

**THE DEVELOPMENT OF INVERSE
FUNCTIONS FOR A REDUNDANT
MANIPULATOR WITH APPLICATIONS
IN JOINT LIMIT AND
OBSTACLE AVOIDANCE**

NAGW-1333

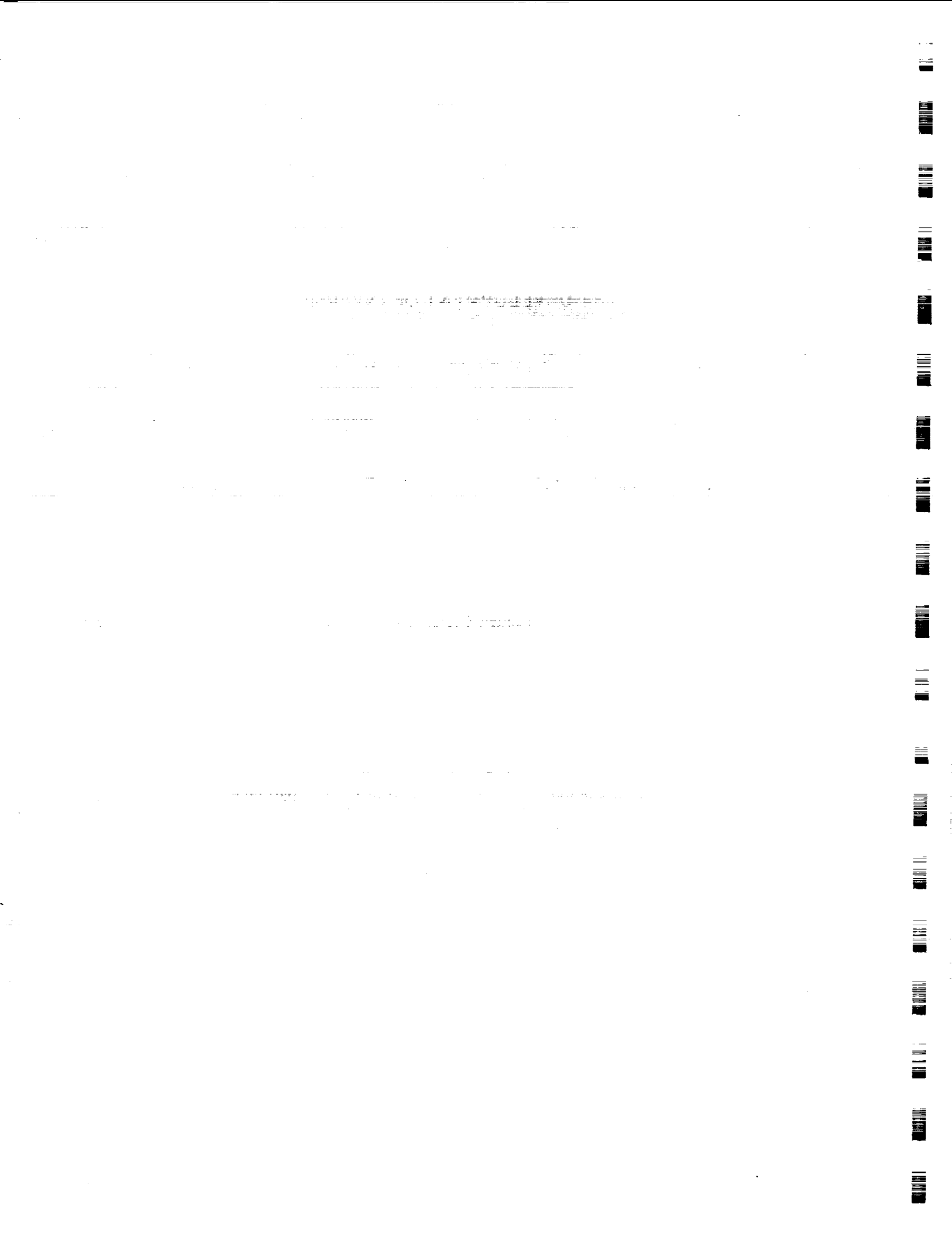
by

Linden Carmichael

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering Department
Troy, New York 12180-3590

May 1992

CIRSE REPORT #116



CONTENTS

LIST OF TABLES	iv
LIST OF FIGURES	v
ACKNOWLEDGMENT	vii
ABSTRACT	viii
1. INTRODUCTION	1
1.1 Motivation	3
1.2 Objectives	4
1.3 Summary	4
2. LITERATURE REVIEW	6
3. FORWARD KINEMATICS	9
3.1 Redundant Task Space	12
3.1.1 Puma-Positioner Configuration variables	13
3.1.2 Puma-Platform Configuration variables	14
3.2 Pose Space	18
3.2.1 Puma-Positioner pose variables	19
3.2.2 Puma-Platform pose variables	23
4. INVERSE KINEMATICS	27
4.1 Puma-Positioner inverse function	28
4.1.1 Calculation of joints 1 - 3	29
4.1.2 Calculation of joints 4 - 6	29
4.1.3 Calculation of the PUMA's wrist joints 7 - 9	32
4.2 Puma-Platform inverse function	34
4.2.1 Calculation of joints with joint four set to zero	34
4.2.2 Rotation of primary elbow by configuration variable θ	38
4.2.3 Calculation of Actual joint values	39

5. EFFECT OF INVERSE FUNCTIONS ON MANIPULATOR SINGULARITIES	43
5.1 Characterization of the CIRSSE manipulator's singularities	44
5.2 The inverse functions and the workspace singularities	45
6. APPLICATION OF PUMA-POSITIONER TO JOINT LIMIT AVOIDANCE	47
6.1 Definition of the decision algorithm	48
6.1.1 Configuration variable q_1 (Joint 1)	49
6.1.2 Configuration variable q_2 (Joint 2)	50
6.1.3 Configuration variable q_3 (Joint 3)	52
7. APPLICATION OF THE PUMA-PLATFORM TO OBSTACLE AVOIDANCE	57
7.1 Link and Obstacle modeling	59
7.2 Generating the closest point of contact	59
7.3 Generation of force fields	62
7.4 Assignment of the Configuration variables	62
7.4.1 Assignment of $d1$	63
7.4.2 Assignment of he	64
7.4.3 Assignment of $theta$	68
7.5 Resolving obstacle conflicts	69
8. TEST AND SIMULATION RESULTS	71
9. CONCLUSION	78
9.1 Future Work	78
APPENDICES	79
A. Symbol Definitions	79
B. Modified Denavit-Hartenberg Representation	80
C. The Jacobian of the CIRSSE manipulator	81
D. The Augmented Jacobian of the CIRSSE manipulator	83
LITERATURE CITED	85

LIST OF TABLES

Table 6.1	Distance constraints	51
Table 8.1	Test results for the inverse functions	72

LIST OF FIGURES

Figure 1.1	Model of CIRSSE Robotic facilities	2
Figure 3.1	Model of CIRSSE (9 dof) manipulator	11
Figure 3.2	Description of Configuration variable <i>he</i>	15
Figure 3.3	Description of Configuration variable <i>theta</i>	17
Figure 3.4	RIGHT and LEFT Arm poses	20
Figure 3.5	Wrist in FLIP and NO-FLIP poses	21
Figure 3.6	Wrist ABOVE and BELOW Secondary elbow	22
Figure 3.7	Secondary elbow ABOVE and BELOW Primary elbow	23
Figure 3.8	Primary elbow in FLIP and NO-FLIP poses	25
Figure 4.1	Model of CIRSSE manipulator with coordinate frames	28
Figure 4.2	Puma-Positioner: Calculation of joint 4	30
Figure 4.3	Puma-Positioner: Calculation of joints 5,6	31
Figure 4.4	Calculation of the PUMA's wrist joints	32
Figure 4.5	Puma-Platform: Calculation of joint 2 (Joint 4 = 0)	35
Figure 4.6	Puma-Platform: Calculation of joints 5,6	36
Figure 4.7	Puma-Platform:Calculation of joint three (Joint 4 = 0)	37
Figure 4.8	Evaluating the pose variable <i>elbow</i>	39
Figure 4.9	Puma-Platform:Calculation of joints 2,3 (Actual values)	40
Figure 4.10	Puma-Platform:Calculation of joint 4 (Actual value)	41
Figure 5.1	Singular configurations of CIRSSE manipulator	45
Figure 6.1	Generating configuration variable q_1	49
Figure 6.2	Joint limit avoidance: Distance constraints	51
Figure 6.3	Range of q_n angles for wrist ABOVE secondary elbow	52

Figure 6.4	Range of qn angles for wrist BELOW secondary elbow	53
Figure 6.5	Description of orientation coordinate frame	54
Figure 6.6	Description of orientation constraints	55
Figure 7.1	Link and Obstacle model	59
Figure 7.2	Description of rotation matrix $R(n, theta)$	60
Figure 7.3	Generation of model based closest point of contact	61
Figure 7.4	Obstacle Avoidance: Definition of $d1$	63
Figure 7.5	Obstacle Avoidance: Definition of he (Link 3)	64
Figure 7.6	Obstacle Avoidance: Definition of he (Link 4)	65
Figure 7.7	Obstacle Avoidance: Definition of he (Link 5)	67
Figure 7.8	Obstacle Avoidance: Definition of $theta$ (Links 3-5)	68
Figure 8.1	Joint-Limit-Avoidance: Wrist BELOW Secondary elbow . . .	73
Figure 8.2	Joint-Limit-Avoidance: Wrist BELOW Secondary elbow . . .	74
Figure 8.3	Joint-Limit-Avoidance: Wrist ABOVE Secondary elbow . . .	74
Figure 8.4	Joint-Limit-Avoidance: Wrist ABOVE Secondary elbow . . .	75
Figure 8.5	Obstacle-Avoidance: Manipulators at time $t = 0.0$	75
Figure 8.6	Obstacle-Avoidance: Manipulators at time $t = 1.0$	76
Figure 8.7	Obstacle-Avoidance: Manipulators at time $t = 2.0$	76
Figure 8.8	Obstacle-Avoidance: Manipulators at time $t = 3.0$	77
Figure 8.9	Obstacle-Avoidance: Manipulators at time $t = 4.0$	77

ACKNOWLEDGMENT

I would like to express by gratitude to Dr. George Saridis whose patience and understanding made it all possible. I would also like to thank Dr. Steve Murphy for his insights and guidance. My parents provided a great deal of support for which I will always be thankful. Finally, I would like to acknowledge the CIRSSE staff and students who were all great to work with.

ABSTRACT

Redundant manipulators greatly enhance the performance of robotic systems over standard non-redundant manipulators. Their larger joint space is capable of meeting user defined task constraints in addition to the standard Cartesian task constraints. However, the key to the efficient performance of these manipulators is in the method of utilizing their increased joint space or the redundancy resolution approach. The redundancy resolution approach taken for the CIRSSE (9 dof) arm involves the development of two inverse functions, the Puma-Positioner and the Puma-Platform. The Puma-Positioner defines a secondary task vector as the position of the base of the PUMA which is then used to expand the workspace of the PUMA. In addition, a joint limit avoidance scheme is generated around the Puma-Positioner that defines internally the secondary task vector and generates a joint vector whose components are within their joint limits. The Puma-Platform defines the secondary task vector as the internal position of the links which is then used to accurately position and orient the end-effector and the manipulator's links. Finally, an obstacle avoidance algorithm is created around the Puma-Platform that defines its secondary task vector based on obstacles entering the manipulator's workspace. This secondary task vector is then used to generate link motion that occurs in the null-space of the end-effector while avoiding the obstacles.



CHAPTER 1

INTRODUCTION

Redundant manipulators have become an integral part of most robotic systems. They increase the functionality of the robotic system by offering extra degrees of freedom and the ability to cope with unexpected disturbances. A manipulator is characterized by the number of its joints which also determine the degree of freedom (dof) of the manipulator's joint space. For a non-redundant manipulator, the dof of the manipulator's joint space is equal to the dof its task space. The dof of the task space is determined by the minimum number of variables that are required to describe any position and orientation in that task space. However, a redundant manipulator has a greater dof in its joint space than its task space. This inequality causes the mapping between the joint space and the task space to be non-unique for the forward mapping and non-existent for the inverse mapping.

Redundant manipulators have been recognized as being superior to standard manipulators due to their increased joint space. The fact that there are now more joints than are needed to meet the constraints of the task space allows for the development of additional constraints. These constraints can be used to increase manipulability, minimize joint torques and as covered in this thesis, avoid joint limits and obstacles. The task and user defined constraints can be formulated into an optimization problem. This optimization or redundancy resolution problem is solved in this thesis through the development of an inverse function. An inverse function generates a one-to-one and cyclic mapping between the task space and the joint space of a redundant manipulator. A cyclic mapping is defined as a mapping in which a closed path always maps into another closed path. The inverse functions augment the task space with a redundant task space; thus adding the necessary extra degrees of freedom to the task space. The redundant task space is spanned by

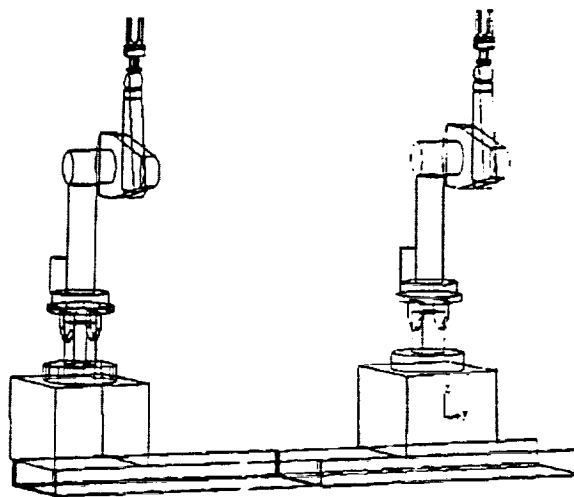


Figure 1.1: Model of CIRSSE Robotic facilities

configuration variables. The definition of these configuration variables differentiates one inverse function from another. Two inverse functions, the Puma-Positioner and the Puma-Platform, are implemented in this thesis. The redundant task space of the Puma-Positioner is described by the three platform joints, q_1 , q_2 and q_3 . These joints, which determine the position of the PUMA's base, are user specified and thus allow the PUMA to be translated to a new location. The redundant task space of the Puma-Platform is described by three spatially visualizable configuration variables, d_1 , h_e and θ_e . These variables describe the location of the links and thus allow the user to accurately specify the desired location of the manipulator's end-effector and its links.

The formulation of the Puma-Positioner and Puma-Platform inverse functions lend themselves to the applications of joint-limit avoidance and obstacle avoidance, respectively. The Puma-Positioner is incorporated into a joint-limit avoidance scheme by utilizing its three configuration variables in such a manner as to meet the position and orientation constraints of the end-effector while avoiding joint limits. The Puma-Platform's three spatially visualizable configuration variables are used

in an obstacle avoidance scheme to reconfigure the the internal configuration of the links in such a manner as to avoid any obstacles without affecting the end-effector's motion.

1.1 Motivation

The CIRSSE facilities currently consist of two PUMA manipulators of the 560 and 600 series each mounted on a three dof platform, as described in figure 1.1. Since each pair of manipulators can be grouped into a single nine dof manipulator, they are considered to be redundant manipulators. Redundant manipulators require special techniques to produce adequate mappings between the joint space and the task space. These techniques should not only meet the position and orientation constraints of the end-effector, but should also utilize the extra joints in an efficient manner. The inverse function approach allows for the efficient usage of the extra joints. Constraints can be easily implemented to meet any user defined needs.

The Puma-Positioner and Puma-Platform inverse functions were developed in order to meet the needs of the lab. A great deal of work is done exclusively on the PUMA. An inverse function was needed that would allow one to not only perform tasks on the PUMA, but also be able to use the platform to reposition the PUMA and thus expand its workspace. This was accomplished through the development of the Puma-Positioner. Also, even with a redundant manipulator, the problem of joint limits still exist. A joint-limit avoidance scheme was needed that would be able to meet Cartesian task constraints and also avoid joint limits. This scheme was developed with the aid of the Puma-Positioner inverse function. The PUMA atop the platform can viewed as a nine dof manipulator. It would be beneficial to have a mapping that would allow the user , as is the case in non-redundant manipulators, to be able to visually identify the configuration of the manipulator. This would require that the redundant task space be parameterized by visualizable

configuration variables. All this was accomplished through the development of the Puma-Platform. Finally, since the two CIRSSE manipulators would be cooperating on tasks and working among obstacles, an obstacle avoidance scheme was needed. This scheme should be fast and should generate internal link motion without affecting the end-effector's motion. Such a scheme was developed through the use of the Puma-Platform.

1.2 Objectives

Motivated by the structure of the CIRSSE facilities, four objectives were determined and met. These objectives are as follows:

- To develop a Puma-Positioner inverse function that would use the platform to greatly expand the workspace of the PUMA.
- To develop a joint-limit avoidance scheme based on the Puma-Positioner that would allow the user access to the expanded workspace of the PUMA without the hassle of hitting joint limits.
- To develop a Puma-Platform inverse function that would model the PUMA and the platform as a single nine dof manipulator and be able to accurately describe the position and orientation of the end-effector and the internal configuration of the links.
- To develop an obstacle avoidance scheme base on the Puma-Platform that would utilize the different internal configurations of the links to avoid obstacles.

1.3 Summary

The CIRSSE facilities consist of two nine dof manipulators. Two inverse functions were developed for these redundant manipulators that would provide one-to-one and cyclic mappings between the joint space and task space of the manipulator.

These inverse functions, the Puma-Positioner and the Puma-Platform, were then respectively developed into joint limit and obstacle avoidance schemes. In the following chapters, a detailed description of the inverse functions and the joint limit and obstacle avoidance schemes will be presented. In chapter two, a literature review of all the material pertaining to this thesis is given. Chapter three deals with the forward kinematics of the redundant manipulator. Chapter four covers the inverse kinematics. The effect of the inverse functions on manipulator singularities is covered in chapter five. Chapters six and seven deal with the implementation of the inverse functions in a joint limit avoidance and obstacle avoidance scheme, respectively. Chapter eight provides the testing and simulation results and chapter nine offers a conclusion.

CHAPTER 2

LITERATURE REVIEW

The redundancy resolution problem for redundant manipulators is an optimization problem that deals with the determination of a joint vector that meets both Cartesian and user defined task constraints. A great deal of research effort has been devoted to redundancy resolution and has resulted in the generation of many different approaches to this problem. These approaches can be divided into two major classes, global and local optimization schemes. The global approaches, as used by Nakamura[1] and Hollerbach[2], generally assume that total knowledge of the tasks and the workspace is available. They are usually iterative and computationally intense. These types of approaches lend themselves more to off-line applications than on-line ones. On-line applications generally utilize local optimization schemes. These schemes can be subdivided into Jacobian based and non-Jacobian based schemes. The Jacobian based schemes translate task and user defined constraints into joint velocities. These joint velocities are then integrated to provide the joint paths. These schemes tend to be computationally intense since they usually require the computation of the generalized inverse of the Jacobian. These schemes also tend to suffer from drifting. Drifting is defined as the case where for a fixed set of constraints, the corresponding joint vector that's produced varies over multiple runs. This is of great importance in assembly lines where accurate repetition of tasks is of vital interest. The different Jacobian based methods vary only in the computation of the joint velocity vector. Whitney[5] utilized the Moore-Penrose Jacobian pseudo-inverse to produce joint velocities. This generalized inverse optimized the square of the joint velocities. Sukhan Lee and Jang Lee[8] decomposed a redundant manipulator into multiple non-redundant manipulators. The generalized inverse of the Jacobian was

used to translate the task constraints into joint velocities for the multiple manipulators. Zghal and Dubey[6,7] introduced an efficient gradient projection optimization scheme for redundant manipulators. By extracting a non-singular matrix from the Jacobian matrix, they were able to avoid calculating the generalized inverse of the Jacobian. Sciavicco and Sicilano[9, 10] introduced the augmented Jacobian method. They augmented the manipulator's Jacobian with enough constraints to generate a square Jacobian matrix. The transpose of the augmented Jacobian was placed in a feedback loop and used to map a Cartesian space error into a joint velocity. This method escaped having to use the generalized inverse of the Jacobian but suffered from convergence problems near singular points.

The non-Jacobian based methods refer mostly to inverse functions. Seraji[11] dealt with some control aspects of inverse functions. Wampler[12] and Shamir[14] provided a comparison between inverse functions and other redundancy resolution approaches. Hollerbach[13] developed an inverse function for a seven dof manipulator. These four authors are the source of most of the material concerning inverse functions that's presented in this thesis. Inverse functions produce one-to-one and cyclic mappings between the joint space and the augmented task space of a manipulator. They tend to be computationally efficient since they don't require the calculation of the Jacobian or its inverse. Inverse functions map tasks directly into the joint space without having to calculate joint velocities and integrating them. Inverse functions also give one some control over singularities as discussed by Bedrossian[15]. The effect of inverse functions on the singularities of the CIRSSE manipulator is presented in chapter five. One possible drawback of the inverse function approach is that it is not a generic process. The manipulator's Jacobian can always be determined and thus Jacobian based schemes apply for all manipulators. However, the inverse function defined for one manipulator will not necessarily work on another. This weakness can also be viewed as a strength. If an inverse function is found

for a manipulator, then this inverse function will be based solely on that manipulator's characteristics. This means that the inverse function will be both fast and efficient. The inverse functions, the Puma-Positioner and the Puma-Platform, presented in this thesis exhibit these characteristics. These inverse functions will now be discussed in some detail.

CHAPTER 3

FORWARD KINEMATICS

The forward kinematics describes the mapping from the joint space of a manipulator to its task space. The joint space consists of all possible joint configurations that the manipulator can assume. The task space contains all the possible position and orientations that the manipulator's end-effector can take. This forward kinematic mapping assumes the form of

$$x = f(q) \tag{3.1}$$

where q is a $(n \times 1)$ joint vector, x a $(m \times 1)$ task vector and f is a non-linear function. In addition to the above mapping, the forward kinematics also includes mappings from the joint space into a set of discrete variables. These variables are termed pose variables and describe the discrete set of joint configurations that correspond to each task vector. For non-redundant manipulators, the mapping from the joint space to the task space and the discrete variables is one-to-one. This fact indicates that the size or degree of freedom (dof) of the joint space, which is determined by the number of manipulator joints, is equal to the dof of the task space. The task space has six dofs for a spatial manipulator where three of those degrees are for positioning and three are for orienting the end-effector. For a redundant manipulator, the joint space is larger than the task space. This means that the forward kinematic mapping is not unique. Also, in addition to the discrete set of joint solutions that correspond to each task vector, there also exists an infinite number of joint solutions for each task vector. There now exists a great deal of flexibility in choosing the joint configurations for a desired task space vector. How one makes this choice is the central theme of redundancy resolution and is addressed

in the development of the inverse functions in this thesis.

The inverse function resolves the redundancy resolution problem by generating a one-to-one forward kinematic mapping. It accomplishes this by augmenting the manipulator's task space with a redundant task space. This redundant task space provides the necessary extra dofs to the task space so that each joint in the joint space vector is independently represented in the augmented task space vector. The augmented forward kinematic mapping takes the form of:

$$\begin{aligned} x &= f(q) \\ x_r &= g(q) \end{aligned} \quad (3.2)$$

where x_r is the ($r=n-m$) redundant task vector and g maps q into x_r . The redundant task space is spanned by r variables termed configuration variables or kinematic functions. These functions can be chosen arbitrarily with the only condition being that they are independent of each other and the position and orientation of the end-effector. Their independence is verified by calculating the augmented Jacobian. The standard manipulator Jacobian maps the joint velocities into the task velocities. The augmented Jacobian maps the joint space velocities into both the task space and redundant task space velocities. It takes the form of:

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{x}_r \end{bmatrix} &= J_a(q) * \dot{q} \\ J_a &= \begin{bmatrix} J_e \\ dg/dq \end{bmatrix} \end{aligned} \quad (3.3)$$

where J_e is the ($m \times n$) manipulator Jacobian, dg/dq is the ($r \times n$) matrix representing the gradient of the ($r \times 1$) vector, g and J_a is the ($n \times n$) augmented Jacobian. The independence of the kinematic functions, g , is determined by evaluating the rank of the augmented Jacobian. If the augmented Jacobian is rank

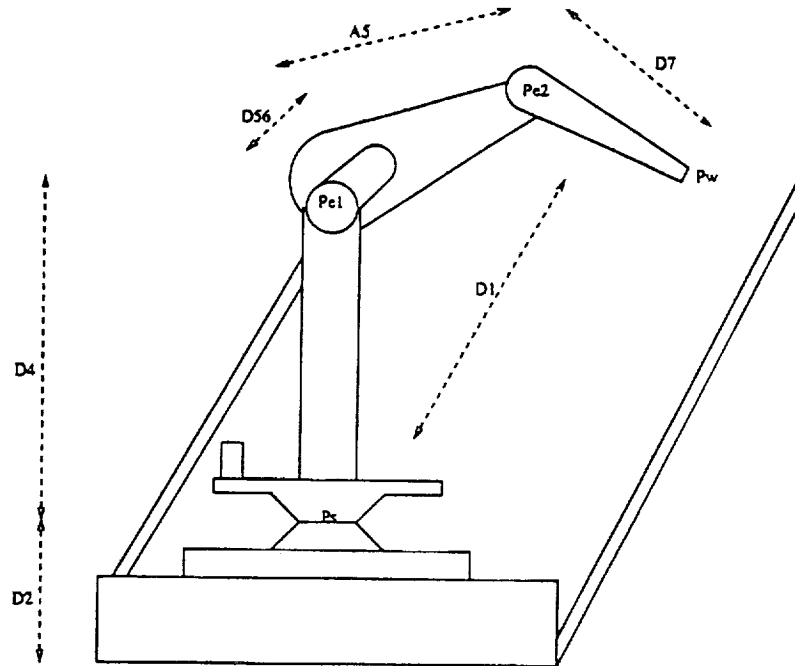


Figure 3.1: Model of CIRSSE (9 dof) manipulator

deficient then the kinematic functions are not independent and new ones must be selected.

The CIRSSE manipulator consists of a six dof PUMA atop a three dof platform. Figure 3.1 provides a model of this arm where P_s represents the shoulder position, $Pe1$ the primary elbow position, $Pe2$ the secondary elbow position and P_w the wrist position. The inverse functions that are developed in this thesis all correspond to the model in figure 3.1. This CIRSSE manipulator is a nine dof spatial arm that operates in a six dof task space. This fact indicates that there are three extra dofs in the joint space. The inverse functions for this manipulator are thus required to have a three dof redundant task space in order to account for the extra joints. The two inverse functions that were developed are termed the Puma-Positioner and the Puma-Platform. The Puma-Positioner has a redundant task space that is spanned by three configuration variables. These variables are the first three platform joint values and are termed q_1 , q_2 and q_3 . The Puma-Platform

has three kinematic functions in its redundant task space. These functions are termed *d1*, *he* and *theta*. A detailed description of these inverse functions and their redundant task space will be given in the following sections.

The inverse functions not only define the redundant task space, but also the pose space. Although the redundant task space provides the necessary extra dofs, there still exists a discrete set of link configurations that the manipulator can be in for each desired augmented task vector. These link configurations are termed poses and comprise the pose space which is spanned by a set of discrete pose variables. The pose variables for the Puma-Positioner are three in number. They are termed *right_left*, *wrist* and *wst_to_e2*. There are five pose variables in the Puma-Platform and are termed *right_left*, *wrist*, *wst_to_e2*, *e2_to_e1* and *elbow*. A complete description of the redundant task space and pose space for each inverse function will now be given.

3.1 Redundant Task Space

The redundant task space can be considered to be the space consisting of all secondary task vectors. The primary task of any manipulator is to position and orient its end-effector. However, for a redundant manipulator, a secondary task needs to be defined in order to account for the extra joints. These secondary task vectors that comprise the redundant task space are a function of the inverse function being used by the redundant manipulator. The Puma-Positioner inverse function was designed with the purpose of expanding the PUMA's workspace. Its redundant task space is spanned by the three platform joints. The secondary task for this inverse function is to position the PUMA's base at the position specified by the user. The Puma-Platform inverse function was designed with the purpose of modeling the PUMA and platform as a nine joint manipulator. Its redundant task space is spanned by three spatially visualizable configuration variables, *d1*, *he*

and θ . These variables serve to describe the internal position of the links. The secondary task for this inverse function is to position the internal links at a position designated by the user. The redundant task space of the Puma-Positioner obviously differs from that of the Puma-Platform. However, there are several characteristics of the redundant task space that are invariant for these inverse functions and for any other inverse function that's developed for the CIRSSE manipulator. The first characteristic is the dimensionality of this space. Since the CIRSSE manipulator has three redundant dofs, the dimension of the redundant task space is three. This fact ensures that all redundant joints will be represented in the redundant task space. The second characteristic is independence. All redundant task spaces are spanned by a basis vector that's independent of the task space vector. This independence ensures that each joint is represented either by the task space or redundant task space vector. Finally, the last characteristic is the existence of a null-space. The null-space of a redundant manipulator is defined as the set of all joint motions for which there is no change in the position and orientation of the end-effector. The basis vectors that span the redundant task space serve to describe the null-space. A detailed description of the configuration variables that define the redundant task space for the Puma-Positioner and Puma-Platform will now be given.

3.1.1 Puma-Positioner Configuration variables

The configuration variables in the Puma-Positioner are the three joints of the platform, q_1 , q_2 and q_3 . They serve only to indicate the position and orientation of the platform which in turn indicates the position and orientation of the PUMA's base. The secondary task being performed by these kinematic functions is to monitor the location of the PUMA's base. Since the inverse function is used to expand the PUMA's workspace, this definition of the kinematic functions is suitable to this task. Since the kinematic functions exactly match the first three joint angles, they

are identity functions. This means that the rank of the augmented Jacobian is equal to that of the manipulator's Jacobian, J_e . This result is documented in the appendix. Since, by definition, the manipulator's Jacobian is always of full rank except at singular points, these three configuration variables are independent and thus, valid choices.

3.1.2 Puma-Platform Configuration variables

The redundant task space for the Puma-Platform is parameterized by three variables, $d1$, he and $theta$. These variables correspond to a base distance, an elbow height and an angle of elbow rotation, respectively. These variables are spatially visualizable; thus allowing them to be approximately identified by sight. Each variable operates in the null-space of the end-effector. This means that each variable describes a subset of the internal link configurations corresponding to a fixed position and orientation of the end-effector. The secondary task being performed by these kinematic functions is to describe the location of the internal links, i.e. the links between the platform's base and the end-effector. The augmented Jacobian, as described in (3.3), is calculated and is determined to be of full rank except at singular configurations. This result is documented in the appendix. Since the augmented Jacobian is not rank deficient, the kinematic functions are independent of each other and the task vector. The internal link motion of the manipulator is termed the self-motion manifold and is completely described by the redundant task vector x_r . The three redundant variables that comprise x_r each serve to describe an independent subspace of the self-motion manifold of the manipulator. In the next few sections, a detailed description of each variable will be given.

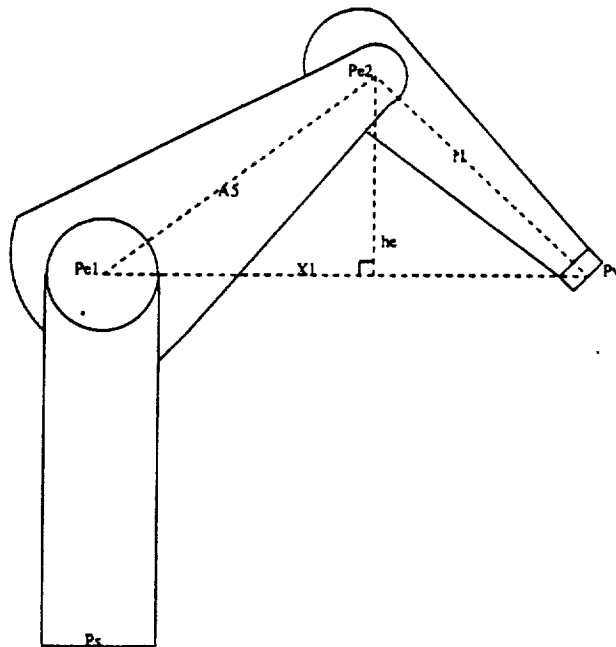


Figure 3.2: Description of Configuration variable h_e

3.1.2.1 Base distance

The base distance d_1 represents the distance from the zeroth coordinate frame to the first coordinate frame located on the base of the platform along the linear rails. This parameter, which is displayed in figure 3.1, was chosen for two reasons. The first being that it was analytically simple since it is actually the first joint of the manipulator. Secondly, by specifying this redundant variable, the user is able to translate the entire manipulator thus changing the approach vector to the target.

3.1.2.2 Elbow height

The elbow height h_e represents the perpendicular distance of the secondary elbow from the line adjoining the primary elbow to the wrist and is shown in figure 3.2. This parameter serves to set the value of joint six and is described by the following set of equations:

$$r1 = \sqrt{A6^2 + D7^2} \quad (3.4)$$

$$x1 = \sqrt{A5^2 + r1^2 + 2 * A5 * (D7 * \sin(q6) + A6 * \cos(q6))} \quad (3.5)$$

$$he = \sqrt{\frac{(2 * A5 * x1)^2 - (x1^2 + A5^2 - r1^2)^2}{4 * x1^2}} \quad (3.6)$$

where $x1$, which represents the distance from the primary elbow to the wrist, and $r1$ represents the total length of link five including the offset $A6$.

It should be noted that due to the difference in lengths of links four and five and the definition of he , $x1$ ranges from $r1 + A5$ to $\sqrt{r1^2 - A5^2}$. This, in turn, causes he to vary between 0 and $A5$. Also, when he is zero, this corresponds to the PUMA elbow singularity. Singularities will be covered in greater detail in the chapter on singularity analysis.

The redundant parameter he was chosen for two reasons. The first being that due the existence of two elbows on this manipulator, only one is needed in order to completely specify motion in the vertical plane. It was necessary to set one of the elbow positions to some specified position. Also, this elbow height serves as a good indicator of where the last two links of the manipulator are. The elbow height was very useful in creating the obstacle avoidance algorithm that is discussed in a later chapter in this thesis.

3.1.2.3 Angle of Elbow rotation

The angle of elbow rotation, θ , is described as the angle of rotation of the primary elbow around the line adjoining the shoulder to the wrist and is shown in figure 3.3. This parameter defines the value of joint four and with the task vector defines the values for joints two and three. The parameter θ , which is a function of the variables n , a unit vector from the shoulder P_s to the wrist P_w , and pe , a unit vector from the shoulder P_s to the primary elbow $Pe1$, can be calculated as

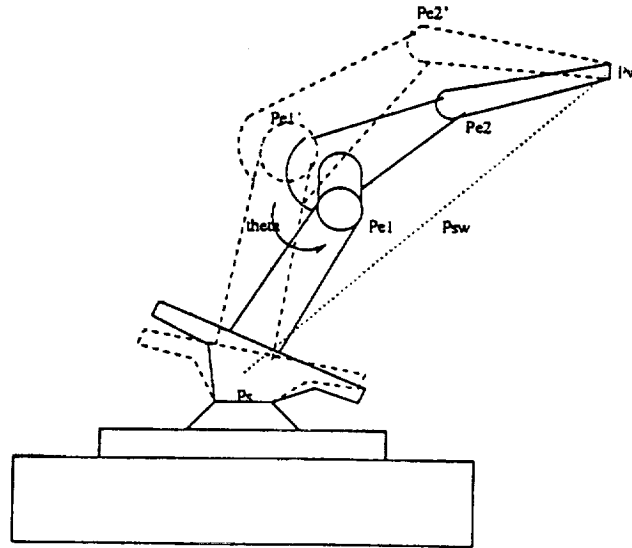


Figure 3.3: Description of Configuration variable θ

follows:

$$pe = \frac{pe1 - ps}{\|pe1 - ps\|} \quad (3.7)$$

$$n = \frac{pw - ps}{\|pw - ps\|} \quad (3.8)$$

$$z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.9)$$

$$\cos(\theta) = \frac{(z \odot n) \odot (pe \otimes n)}{\|z \odot n\| * \|pe \otimes n\|} \quad (3.10)$$

$$\sin(\theta) = \sqrt{1 - \cos^2(\theta)} \quad (3.11)$$

$$\theta = \tan^{-1}\left(\frac{\sin(\theta)}{\cos(\theta)}\right) \quad (3.12)$$

The variable θ thus represents the angle of rotation of the vector pe into the vector z around the unit vector n .

The redundant variable θ models the redundancy inherent in the human arm. The human arm is capable of generating redundant motion by rotating the elbow while keeping the wrist and shoulder fixed. Since the nine joint manipulator, described in figure 3.1, is a model of the human arm, the choice of θ as a redundant parameter is a suitable one. Another reason for choosing θ is that it parameterizes the internal links into a plane rotating around the unit vector n . This parameterization is of immense value in the obstacle avoidance algorithm that's described in a later chapter.

In summary, the redundant task space is parameterized into three separate subspaces by its three configuration variables. This means that each configuration variable is independent of each other. Any variation in one of the variables has no effect on any of the others. The configuration variables for the Puma-Positioner and Puma-Platform inverse functions serve to describe the position of the PUMA's base and the position of the internal links, respectively. These variables can assume a continuous range of values within the manipulator's workspace and allow for the generation of an infinite number of link configurations. The fact that the redundant task space is spanned by continuous variables is one of the major difference between it and the pose space. The pose space is spanned by discrete variables that allow for the generation of a finite number of link configurations.

3.2 Pose Space

The pose space is a characteristic of both redundant and non-redundant manipulators. All rigid link manipulators have a finite number of different link configurations that correspond to a fixed position and orientation. These different link configurations are termed poses and comprise the pose space. The pose space is spanned by a set of parameters termed pose variables. These pose variables are discrete variables that can only assume one of three values. These values are one,

negative one and zero, where zero indicates that the links are aligned. Although both the pose space and the redundant task space represent link configurations, they can't be combined into one space for several reasons. The first is, as stated above, the pose space is spanned by discrete variables while the redundant task space is spanned by continuous variables. Secondly, the redundant task space resides in the null-space of the end-effector, but the pose space doesn't. As one induces motion in the redundant task space, the end-effector remains fixed while the links change configuration. However, as one goes from one pose to another, the end-effector undergoes some motion before returning to its original position and orientation. A thorough description of the pose variables for both inverse functions will now be given.

3.2.1 Puma-Positioner pose variables

The pose space for the Puma-positioner is described by three pose variables, *right_left*, *wrist* and *wst_to_e2*. which describe the poses of the PUMA. It should be noted that since the redundant variables for this inverse function are the three joint angles of the platform, these variables serve also to describe the pose of the platform. Since each pose variable can assume three distinct values, the pose space contains 27 different poses.

3.2.1.1 RightToLeft Arm Indicator

The CIRSSE arm has a link offset located on the shoulder of the Puma attachment as described in figure 3.1 This shoulder offset leads to the definition of left and right arm configurations. The *right_left* configuration variable is described as being either left or right depending on which of ones arms is along the shoulder offset when one is standing on the base of the PUMA and facing the end effector. An example of LEFT and RIGHT arm configurations is shown in figure 3.4 The

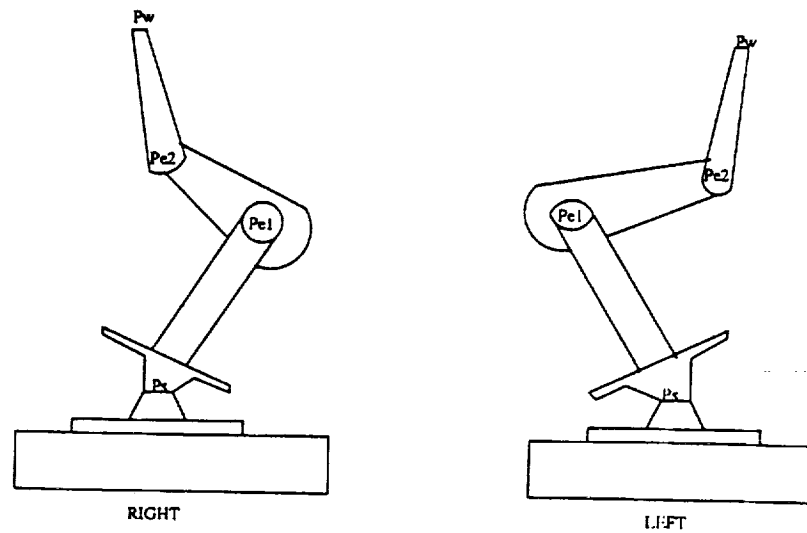


Figure 3.4: RIGHT and LEFT Arm poses

right_left descriptor is described by the following equation:

$$\begin{aligned}
 x1 &= A5 * \cos(q5) + A6 * \cos(q5 + q6) + D7 * \sin(q5 + q6) \\
 right_left &= \text{sgn}(x1)
 \end{aligned} \tag{3.13}$$

where $x1$ represents the distance from the primary elbow $Pe1$ to the wrist, Pw relative to frame four. It should be noted that when the wrist is aligned with the primary elbow, the arm is in an alignment configuration and $x1$ is zero. In this case, the variable *right_left* is defaulted to the left configuration. Another point of interest is that when *right_left* is zero, this signifies the PUMA's shoulder singularity.

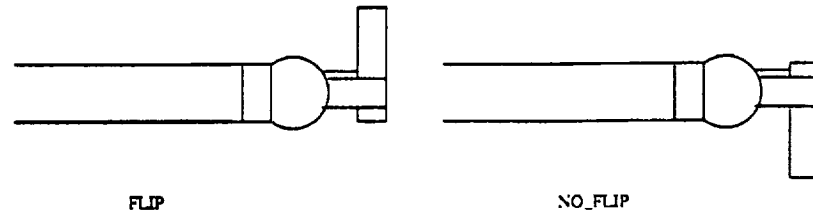


Figure 3.5: Wrist in FLIP and NO-FLIP poses

3.2.1.2 Wrist Indicator

The wrist of the PUMA attachment represents a spherical wrist. There are three joints associated with the wrist; thus allowing the wrist to assume any orientation within its joint constraints. Due to the placement of the three joints, there are two possible solutions for each orientation. The first solution orients the wrist with a positive joint eight while the second solution utilizes a negative joint eight. These solutions are shown in figure 3.5 where a FLIP solution represents a negative joint eight. The *wrist* indicator is used to determine which solution is to be found and is described by:

$$wrist = \text{sgn}(q8) \quad (3.14)$$

The point at which joint eight is zero is a singular point and is the PUMA wrist singularity.

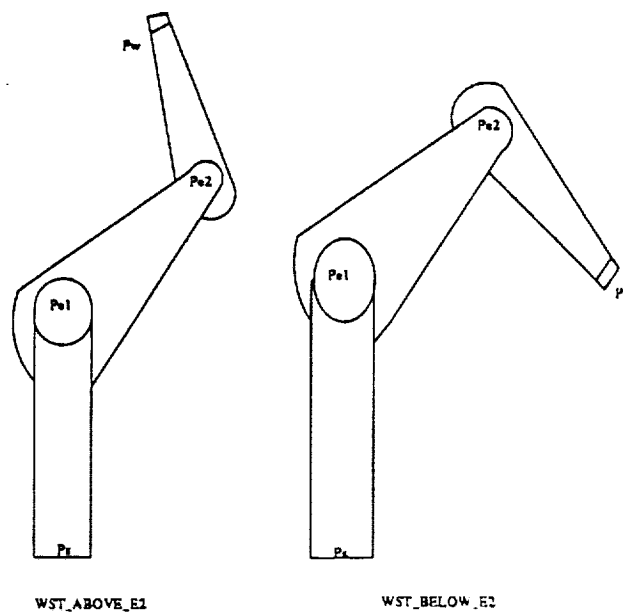


Figure 3.6: Wrist ABOVE and BELOW Secondary elbow

3.2.1.3 Secondary Elbow Indicator

The PUMA's elbow can be either above or below its wrist for a given end-effector position. These dual solutions allow for the existence of the pose variable, wst_to_e2 , which is one or negative one if the wrist is to the left or right of the line adjoining the PUMA's shoulder to its wrist. An example of these two configurations is shown in figure 3.6. The wst_to_e2 indicator is described by:

$$wst_to_e2 = \text{sgn}(q6_{s1} - q6) * \text{sgn}(q6 - q6_{s2}) \quad (3.15)$$

$$q6_{s1} = 92.3445 \text{ deg}$$

$$q6_{s2} = 272.3445 \text{ deg}$$

The pose variable wst_to_e2 assumes a value of zero when the wrist is aligned with the PUMA's elbow. This position is also known as the PUMA elbow singularity.

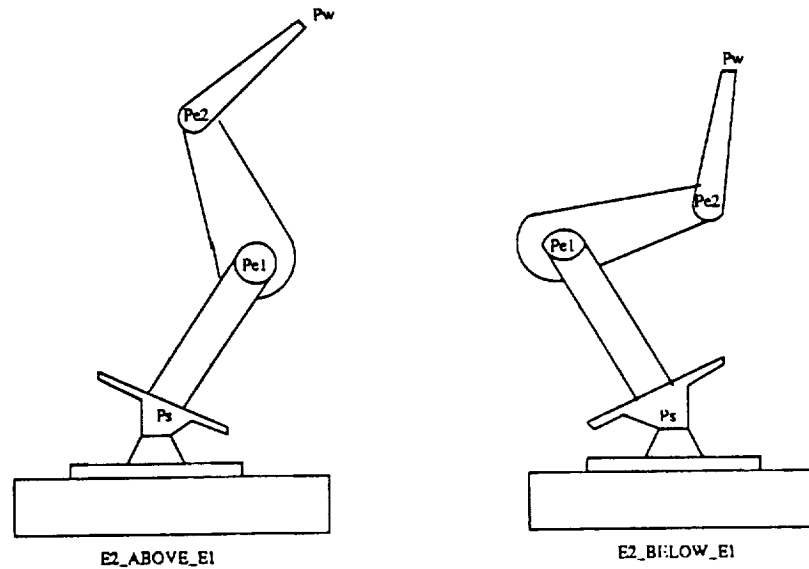


Figure 3.7: Secondary elbow ABOVE and BELOW Primary elbow

3.2.2 Puma-Platform pose variables

The pose space for the Puma-platform is described by five pose variables, *right_left*, *wrist*, *wst_to_e2*, *e2_to_e1* and *elbow*. Since each one of these variables can assume three distinct values, there are 729 poses in this pose space. It turns out that there is some duplication of pose variable description between the Puma-Positioner and the Puma-Platform. The description of the Puma-Platform's pose variables, *right_left*, *wrist* and *wst_to_e2* is given in the previous section. These pose variables only describe the PUMA section of the manipulator. The pose variables *e2_to_e1* and *elbow* describe the different configurations of the platform and its connection to the PUMA. These variables will now be described in greater detail.

3.2.2.1 Primary Elbow Indicator

The primary elbow can be to the right or left of the secondary elbow for any position and orientation of the end effector. Due to the existence of these dual

solutions, the pose variable, $e2_to_e1$, describing the position of primary elbow was necessary. The variable, $e2_to_e1$, is one or negative one if the primary elbow is to the left or right of the line adjoining the secondary elbow to the wrist. The two link configurations imposed by this variable are displayed in figure 3.7. The pose variable, $e2_to_e1$, is described by:

$$e2_to_e1 = \text{sgn}(q5 - 90) * \text{sgn}(270 - q5) \quad (3.16)$$

Although the pose variable, $e2_to_e1$, describes the position of the primary elbow with respect to the secondary elbow, it is not an independent variable. Due to the formulation of the redundant parameters, the variable he describes the position of the primary elbow relative to the secondary elbow. This fact forces the

variable, $e2_to_e1$, into a purely descriptive role. It can describe the configuration of the manipulator's primary elbow. However, to place the manipulator into a certain $e2_to_e1$ configuration, the variable he has to be above or below a certain threshold value. This threshold value corresponds to the value of he when the primary and secondary elbows are aligned and is described by:

$$x1 = \sqrt{A5 * (\|Pw - Ps\|^2 - D4^2) + D4 * (r1^2 - A5^2)} \quad (3.17)$$

$$he^{thr} = \sqrt{\frac{(2 * A5 * x1)^2 - (x1^2 + A5^2 - r1^2)^2}{4 * x1^2}} \quad (3.18)$$

$$e2_to_e1 = \text{sgn}(he - he^{thr}) \quad (3.19)$$

3.2.2.2 Elbow Indicator

The position of the primary elbow is controlled by two joints, joints two and three, which are aligned in such a manner as to cause the existence of two solutions for each position of the primary elbow. These solutions are described by the pose

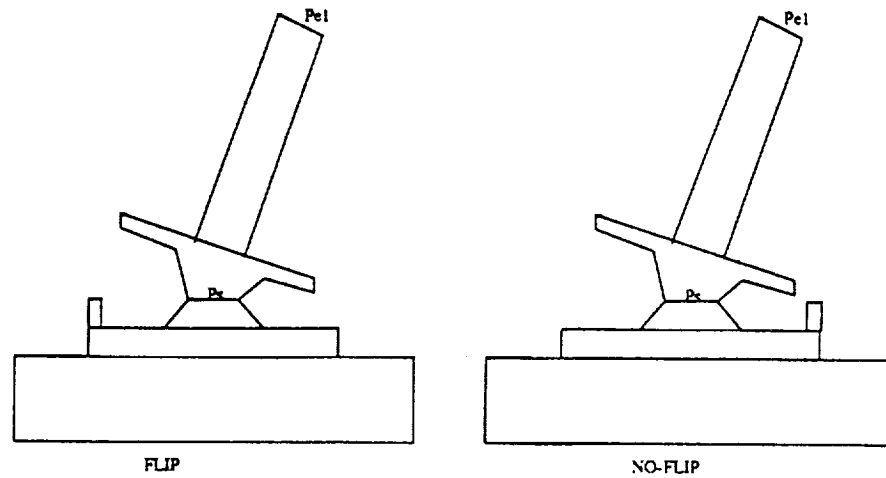


Figure 3.8: Primary elbow in FLIP and NO-FLIP poses

variable, *elbow*, which determines whether joint three is negative or not. An example of these solutions is displayed in figure 3.8 where a FLIP solution indicates a negative joint three. The variable *elbow* is described by:

$$elbow = \text{sgn}(q_3) \quad (3.20)$$

where the point at which joint three is zero is an algorithmic singularity and is described in chapter five. The pose variable *elbow* is another pose variable that serves only a descriptive role in the inverse function. However, this is not due to an overlap between the redundant task space and the pose space, as was the previous case. Instead, this pose variable was defined inside the inverse function in order to avoid the 180 degree joint shift that occurred whenever the algorithmic singular point ($q_3 = 0$) was crossed. The pose variable *elbow* was defined internally as:

$$elbow = \text{sgn}(Pe1_x * Psw_x + Pe1_y * Psw_y) \quad (3.21)$$

and assumes the value of negative one whenever the primary elbow and wrist are in different quadrants.

In summary, the forward kinematics provides the mapping from the joint space of the CIRSSE manipulator to its task space. Its task space is described by a task space vector which consists of a three dimensional positional vector and three orientation vectors. Since the CIRSSE manipulator is a redundant (9 dof) manipulator, the task space was augmented with a 3-dimensional redundant task space. This augmentation ensured that each joint in the joint space was independently represented in the task space. The redundant task space was defined separately by each one of the inverse functions, the Puma-Positioner and the Puma-Platform. The augmented task space along with the necessary pose variables ensured that a one-to-one and cyclic mapping was generated between the joint space and the augmented task space. This one-to-one mapping proves to be crucial in the generation of the inverse mapping, the inverse kinematics.

CHAPTER 4

INVERSE KINEMATICS

The inverse kinematics can be described as the mapping from the task space of the manipulator to its joint space. This mapping is the inverse of the mapping described in (3.1) and takes the form of:

$$q = f^{-1}(x) \quad (4.1)$$

where q is a $(n \times 1)$ joint vector, x a $(m \times 1)$ task vector and f^{-1} is a non-linear function. The above mapping generates a set of joint vectors for each task vector. For non-redundant manipulators, the set of joint vectors that is generated is discrete. Pose variables need to be specified in order to select a unique joint vector from the set. However for redundant manipulators, an infinite set is generated by this mapping. Even with the pose variables, an infinite number of joint vectors can be chosen for each given task vector. As indicated in chapter three, the inverse function solves this problem by determining an additional set of constraints that are termed the redundant task space. These constraints along with the task vector and pose variables allow one to select a unique joint vector from the infinite number of joint vectors generated. Figure 4.1 describes a model of the CIRSSE manipulator with its associated coordinate frames. The coordinate frames are defined under the modified Denavit-Hartenberg representation which is described in the appendix. The two inverse functions, the Puma-Positioner and Puma-Platform, were developed based on this model. These inverse functions both utilize a redundant task space and pose space, as described in chapter three, to generate the joint space vector. The basic process is to request a task space vector, redundant task space vector and pose variables from the user and to then generate the corresponding joint vector. In

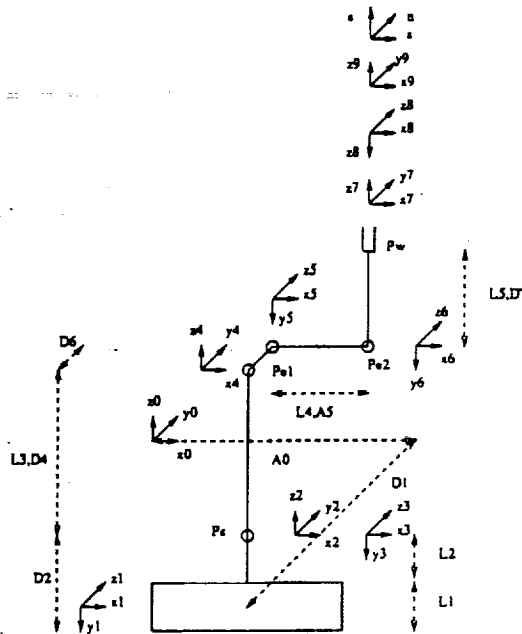


Figure 4.1: Model of CIRSSE manipulator with coordinate frames

the ensuing sections, a detailed description will be given of the inverse mapping of the inverse functions, the Puma-Positioner and the Puma-Platform.

4.1 Puma-Positioner inverse function

The Puma-Positioner models the nine dof CIRSSE manipulator, displayed in figure 4.1, as a PUMA manipulator with a moving base. The position of the platform is specified by the user through the configuration variables, q_1, q_2 and q_3 . This position along with the desired position and orientation of the end-effector and the pose variables is then used to solve for the remaining six joints. The pose variables are described by:

$$right_left = \begin{pmatrix} 1 & LEFT \\ -1 & RIGHT \end{pmatrix} \quad (4.2)$$

$$wrist = \begin{pmatrix} 1 & NO - FLIP \\ -1 & FLIP \end{pmatrix} \quad (4.3)$$

$$wst_to_e2 = \begin{pmatrix} 1 & ABOVE \\ -1 & BELOW \end{pmatrix} \quad (4.4)$$

4.1.1 Calculation of joints 1 - 3

The values of joints one, two and three are just the values of the three configuration variables specified by the user, q_1 , q_2 and q_3 . The position of the primary elbow is then defined as:

$$Pe1 = \begin{bmatrix} A0 + D4 * \cos(q_2) * \sin(q_3) \\ q_1 + D4 * \sin(q_2) * \sin(q_3) \\ D2 + D4 * \cos(q_3) \end{bmatrix} \quad (4.5)$$

Once the position of the primary elbow is known, the task space vector can be recomputed relative to the coordinate frame attached to the primary elbow. This new task space vector along with the pose variables is used to compute joints four through nine.

4.1.2 Calculation of joints 4 - 6

The first step is to calculate the task vector relative to frame three. This is calculated by:

$$T_3^0 = T_3^0(q_1, q_2, q_3) \quad (4.6)$$

$$T_9^3 = T_0^3 * T_9^0 \quad (4.7)$$

$$T_9^3 = \begin{bmatrix} R_9^3 & P_9^3 \\ 00 & 01 \end{bmatrix} \quad (4.8)$$

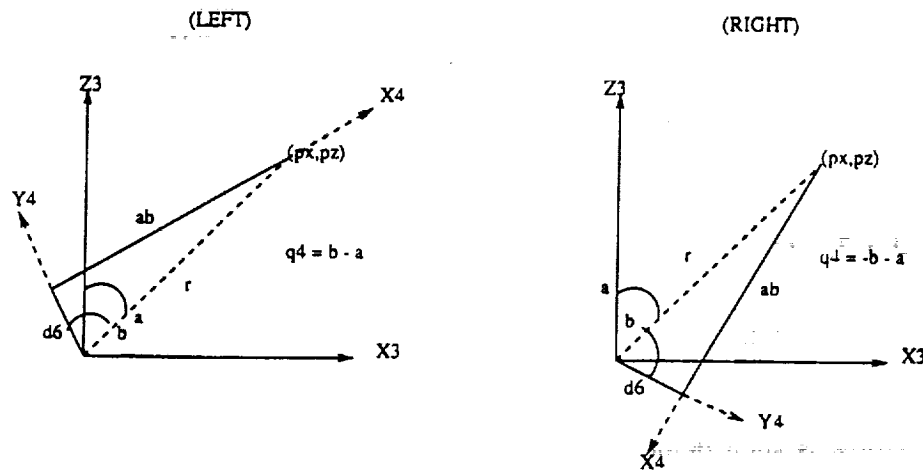


Figure 4.2: Puma-Positioner: Calculation of joint 4

where T_9^0 represents the task relative to frame zero and T_9^3 represents the task relative to frame three. This information is then used to calculate joints four, five and six.

4.1.2.1 Joint 4

Referring to figure 4.2, joint four of the manipulator reflects the two pose configurations. LEFT and RIGHT that the PUMA can assume. The solution is obtained by a projection of the position vector P_{sw} unto the $X^3 - Z^3$ plane.

$$r = \sqrt{px^2 + pz^2} \quad (4.9)$$

$$ab = \sqrt{r^2 - \bar{D}_6^2} \quad (4.10)$$

$$\sin(q_4^l) = \sin(b - a) \quad (4.11)$$

$$\cos(q_4^l) = \cos(b - a) \quad (4.12)$$

$$\sin(q_4^r) = \sin(-b - a) \quad (4.13)$$

$$\cos(q_4^r) = \cos(-b - a) \quad (4.14)$$

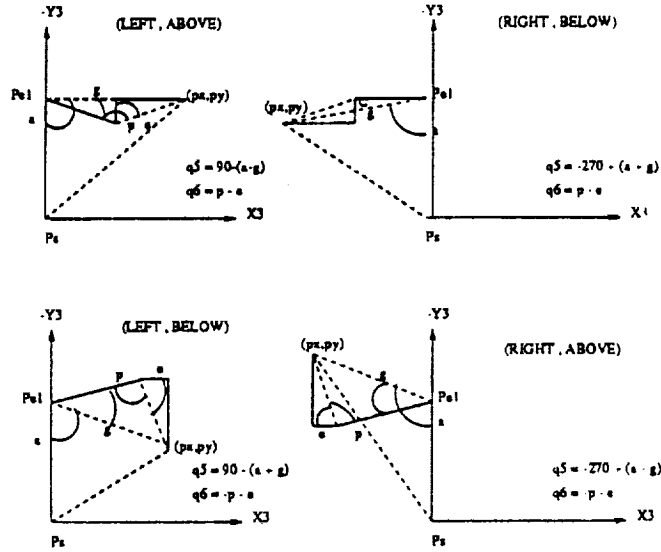


Figure 4.3: Puma-Positioner: Calculation of joints 5,6

Equations 4.11-4.14 can be combined by using the pose variable *right_left* to solve for q_4 .

$$q_4 = \tan^{-1} \left(\frac{-px * \bar{D}6 + right_left * pz * ab}{pz * \bar{D}6 + right_left * px * ab} \right) \quad (4.15)$$

4.1.2.2 Joints 5 and 6

Joints five and six of the manipulator reflect the multiple solutions associated with the pose variables, *right_left* and *wst_to_e2*. Referring to figure 4.3, joints five and six are calculated by projecting the position vector P_{sw} unto the $X^3 - Y^3$ plane and then using the pose descriptors that are defined in (4.2-4.4). Joints five and six are calculated by:

$$r1 = \sqrt{ab^2 + (-py - l4)^2} \quad (4.16)$$

$$\cos(g) = \frac{A5^2 + r1^2 - l1^2}{2 * A5 * r1} \quad (4.17)$$

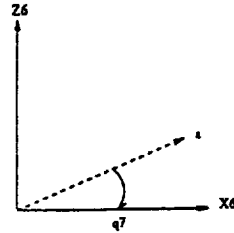


Figure 4.4a

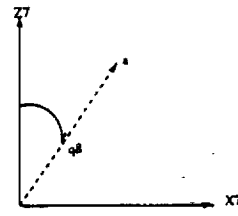


Figure 4.4b

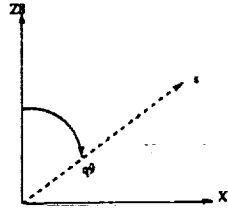


Figure 4.4c

Figure 4.4: Calculation of the PUMA's wrist joints

$$\sin(g) = \sqrt{1 - \cos^2(g)} \quad (4.18)$$

$$\cos(a) = \frac{x1^2 + D4^2 - (ab^2 + py^2)}{2 * x1 * D4} \quad (4.19)$$

$$\sin(a) = \text{right_left} * \sqrt{1 - \cos^2(a)} \quad (4.20)$$

$$\cos(q_5) = \cos(g) * \cos(a) + \text{right_left} * \sin(g) * \sin(a) \quad (4.21)$$

$$\sin(q_5) = \sin(g) * \cos(a) - \text{right_left} * \cos(g) * \sin(a) \quad (4.22)$$

$$q_5 = \tan^{-1}\left(\frac{\sin(q_5)}{\cos(q_5)}\right) \quad (4.23)$$

$$\cos(p) = \frac{A5^2 + r1^2 - x1^2}{2 * A5 * r1} \quad (4.24)$$

$$\sin(p) = \text{right_left} * \sqrt{1 - \cos^2(p)} \quad (4.25)$$

$$q_6 = \tan^{-1}\left(\frac{-\sin(p) * A6 - \cos(p) * D7}{-\cos(p) * A6 + \sin(p) * D7}\right) \quad (4.26)$$

4.1.3 Calculation of the PUMA's wrist joints 7 - 9

The PUMA's wrist joints are used to place the end-effector into the desired orientation. The desired orientation is described by the vectors n , s and a , as described in figure 4.1. Referring to figure 4.1, vector a represents the approach

vector, vector s the sliding vector and n is the normal vector representing the cross product of s and a . Using these orientation vectors and

$$R_6^0(q_1..q_6) = \begin{bmatrix} X^6 & Y^6 & Z^6 \end{bmatrix} \quad (4.27)$$

joints seven, eight and nine will now be solved.

4.1.3.1 Joint 7

Referring to figure 4.1, joint seven is defined as how much the wrist must rotate in order for joint eight to align Z^8 with a . Figure 4.4a describes the FLIP and NO-FLIP configurations for the wrist where the solution for joint seven is defined by:

$$q_7 = \tan^{-1}\left(\frac{wrist * (a \odot Z^6)}{wrist * (a \odot X^6)}\right) \quad (4.28)$$

4.1.3.2 Joint 8

Referring to figure 4.1, joint eight is defined as the amount of tilt that is necessary in the wrist in order for Z^8 to be aligned with a . Figure 4.4b describes the FLIP and NO-FLIP configurations for the wrist where the solution for joint eight is defined by:

$$q_8 = \tan^{-1}\left(\frac{wrist * (a \odot X^7)}{a \odot Z^7}\right) \quad (4.29)$$

4.1.3.3 Joint 9

Referring to figure 4.1, joint nine is defined as the amount of rotation necessary to align s with Y^9 . Figure 4.4c describes the FLIP and NO-FLIP configurations for the wrist where the solution for joint nine is defined by:

$$q_9 = \tan^{-1}\left(\frac{-wrist * (o \odot X^8)}{wrist * (o \odot Z^8)}\right) \quad (4.30)$$

4.2 Puma-Platform inverse function

The Puma-Platform models the PUMA and platform as a nine joint manipulator. The first six joints of the manipulator are used to position the end-effector and place the links into some desired configuration. The constraints that define these joints are the position vector and the three configuration variables. The last three joints are used to orient the end-effector and are defined by the three orientation vectors n, s and a . These three joints are the PUMA's wrist joints and are defined in the previous section. The process for calculating the first six joints can be divided into two steps. In the first step, joint four is set to zero and the remaining joints are calculated. This step defines the unrotated position of the primary elbow, $Pe1$. In the second step, the primary elbow is rotated by $theta$ radians and then joints two through four are calculated. A detailed description of the calculation of the first six joints using the position and configuration constraints will now be given.

4.2.1 Calculation of joints with joint four set to zero

Joints one through six are calculated with joint four set to zero. These joint values correspond to a zero degree rotation of the primary elbow by $theta$. However, since joints one, five and six are not influenced by $theta$, their values calculated in this section represent their actual values.

4.2.1.1 Joint 1

Joint one takes the value of the configuration variable $d1$ and represents the actual value of joint one since it's not affected by the configuration variables, he and $theta$. It is described by:

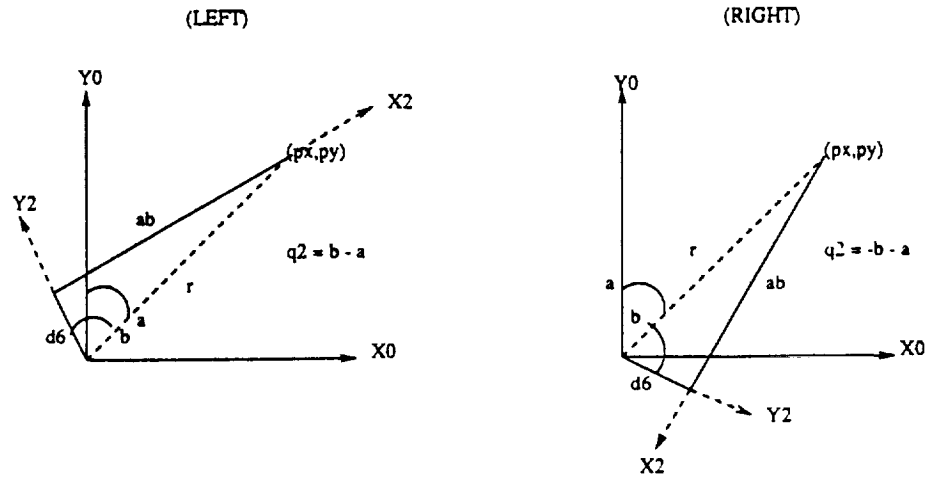


Figure 4.5: Puma-Platform: Calculation of joint 2 (Joint 4 = 0)

$$q_1 = d_1 \quad (4.31)$$

4.2.1.2 Joint 2

Referring to figure 4.5, joint four of the manipulator reflects the two pose configurations, LEFT and RIGHT that the PUMA can assume. The solution is obtained by a projection of the position vector Psw unto the $X^0 - Y^0$ plane.

$$r = \sqrt{px^2 + py^2} \quad (4.32)$$

$$ab = \sqrt{r^2 - \bar{D}_6^2} \quad (4.33)$$

$$\sin(q_2^l) = \sin(b - a) \quad (4.34)$$

$$\cos(q_2^l) = \cos(b - a) \quad (4.35)$$

$$\sin(q_2^r) = \sin(-b - a) \quad (4.36)$$

$$\cos(q_2^r) = \cos(-b - a) \quad (4.37)$$

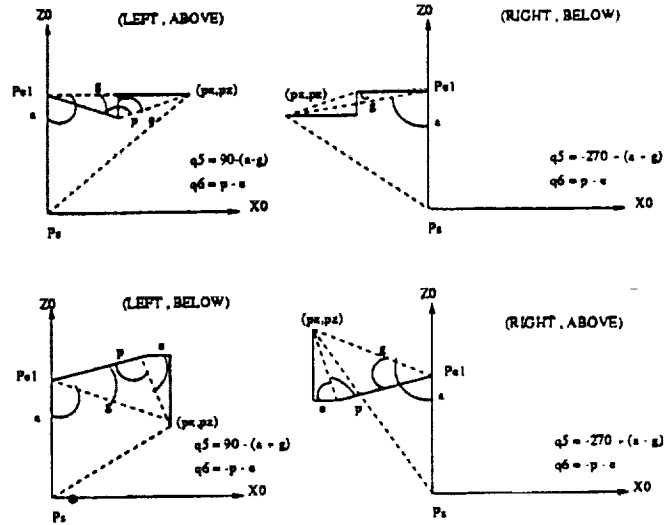


Figure 4.6: Puma-Platform: Calculation of joints 5,6

Equations 4.34-4.37 can be combined by using the pose variable *right_left*, which is defined in (4.2) to solve for q_2 .

$$q_2 = \tan^{-1} \left(\frac{-px * \bar{D}6 + right_left * py * ab}{py * \bar{D}6 + right_left * px * ab} \right) \quad (4.38)$$

4.2.1.3 Joints 5 and 6

Joints five and six of the manipulator reflect the multiple solutions associated with the pose variables, *right_left* and *wtl_to_e2*. Referring to figure 4.6, joints five and six are calculated by projecting the position vector Psw unto the $X^0 - Z^0$ plane and then using the pose descriptors that are defined in (4.2-4.4). Joints five and six are calculated by:

$$r1 = \sqrt{A5^2 - h\epsilon^2} + \sqrt{r1^2 - h\epsilon^2} \quad (4.39)$$

$$\cos(g) = \frac{A5^2 + r1^2 - r1^2}{2 * A5 * r1} \quad (4.40)$$

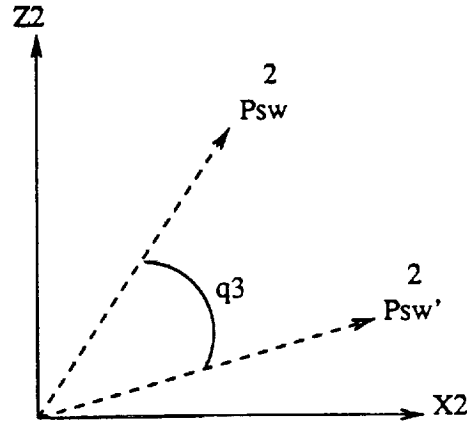


Figure 4.7: Puma-Platform: Calculation of joint three (Joint 4 = 0)

$$\sin(g) = \sqrt{1 - \cos^2(g)} \quad (4.41)$$

$$\cos(a) = \frac{r1^2 + D4^2 - (ab^2 + pz^2)}{2 * r1 * D4} \quad (4.42)$$

$$\sin(a) = right_left * \sqrt{1 - \cos^2(a)} \quad (4.43)$$

$$\cos(q_5) = \cos(g) * \cos(a) + right_left * \sin(g) * \sin(a) \quad (4.44)$$

$$\sin(q_5) = \sin(g) * \cos(a) - right_left * \cos(g) * \sin(a) \quad (4.45)$$

$$q_5 = \tan^{-1}\left(\frac{\sin(q_5)}{\cos(q_5)}\right) \quad (4.46)$$

$$\cos(p) = \frac{A5^2 + r1^2 - x1^2}{2 * A5 * r1} \quad (4.47)$$

$$\sin(p) = right_left * \sqrt{1 - \cos^2(p)} \quad (4.48)$$

$$q_6 = \tan^{-1}\left(\frac{-\sin(p) * A6 - \cos(p) * D7}{-\cos(p) * A6 + \sin(p) * D7}\right) \quad (4.49)$$

4.2.1.4 Joint3

Referring to figure 4.7, joint three is calculated with joint four set to zero by projecting the position vector Psw into the $X^2 - Z^2$ plane. Joint three is defined by:

$$P_9^2 = R_0^2 * P_{sw}^0 \quad (4.50)$$

$$P2x' = A5 * \cos(q_5) + A6 * \cos(q_5 + q_6) + D7 * \sin(q_5 + q_6) \quad (4.51)$$

$$P2z' = D4 - A5 * \sin(q_5) - A6 * \sin(q_5 + q_6) + D7 * \cos(q_5 + q_6) \quad (4.52)$$

$$q_3 = \tan^{-1} \left(\frac{P2z' * P_{9x}^2 - P2x' * P_{9z}^2}{P2x' * P_{9x}^2 + P2z' * P_{9z}^2} \right) \quad (4.53)$$

4.2.2 Rotation of primary elbow by configuration variable θ

The position of the unrotated primary elbow can be calculated by:

$$Pe' = \begin{bmatrix} D4 * \cos(q_2) * \sin(q_3) \\ D4 * \sin(q_2) * \sin(q_3) \\ D4 * \cos(q_3) \end{bmatrix} \quad (4.54)$$

Due to the PUMA's shoulder offset, the position of the unrotated primary elbow doesn't correspond to the zero position for θ . In order to compensate for this offset, the rotation induced by this offset has to be subtracted from θ . This subtraction will rotate the primary elbow by the desired amount. The primary elbow rotation induced by the shoulder offset is termed θ' and is calculated as follows:

$$n = \frac{P_{sw}}{\|P_{sw}\|} \quad (4.55)$$

$$zCn = \begin{bmatrix} \frac{-ny}{\sqrt{nx^2 + ny^2}} \\ \frac{nx}{\sqrt{nx^2 + ny^2}} \\ 0 \end{bmatrix} \quad (4.56)$$

$$peCn = \frac{Pe' \odot n}{\|Pe' \odot n\|} \quad (4.57)$$

$$\theta' = \tan^{-1} \left(\frac{\sqrt{1 - (zCn \odot peCn)^2}}{zCn \odot peCn} \right) \quad (4.58)$$

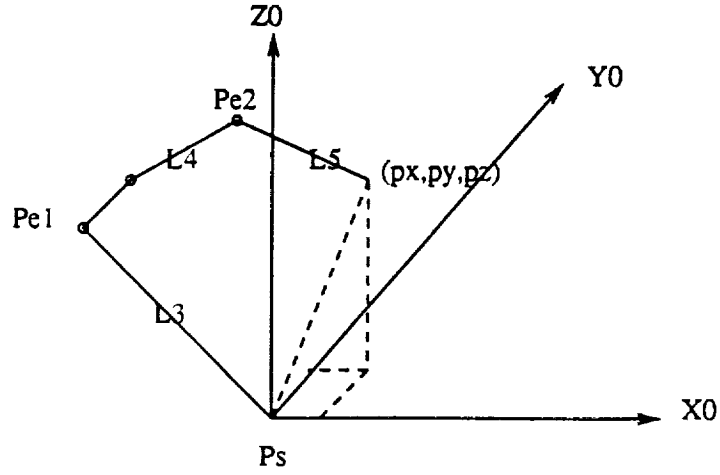


Figure 4.8: Evaluating the pose variable *elbow*

Once θ' is calculated, the new value for θ becomes

$$\theta = \theta - \theta' \quad (4.59)$$

This value for θ is then used to rotate the primary elbow around n . The new rotated position of the primary elbow is determined by:

$$Pe = n \otimes Pe' * (1 - \cos(\theta)) + Pe' * \cos(\theta) + n \otimes Pe' * \sin(\theta) \quad (4.60)$$

4.2.3 Calculation of Actual joint values

Once Pe is calculated, the next step is to calculate the pose variable *elbow*. This pose variable determines whether joint three is positive or negative. Referring to figure 4.8, it should be noted that this manipulator's configuration reflects the case when the position of the primary elbow and the position of the end-effector are

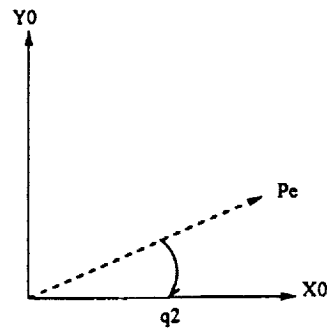


Figure 4.9a

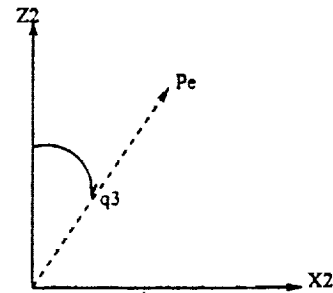


Figure 4.9b

Figure 4.9: Puma-Platform: Calculation of joints 2,3 (Actual values)

in two different quadrants. In this case, joint three assumes a negative value and the pose variable *elbow* is set to negative one. The pose variable *elbow* is described by:

$$elbow = \text{sgn}(Pe_x * px + Pe_y * py) \quad (4.61)$$

Given the position of the rotated primary elbow *Pe* and the value of the pose variable, *elbow*, the actual values for joints two, three and four can now be calculated.

4.2.3.1 Joint 2

Referring to figure 4.9a, joint two is calculated by:

$$q_2 = \tan^{-1}\left(\frac{elbow * Pe_y}{elbow * Pe_x}\right) \quad (4.62)$$

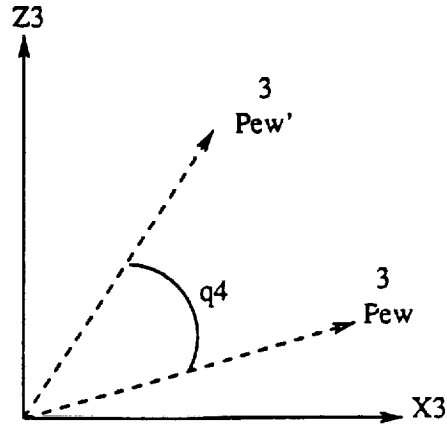


Figure 4.10: Puma-Platform: Calculation of joint 4 (Actual value)

4.2.3.2 Joint 3

Referring to figure 4.9b, joint three is calculated by:

$$q_3 = \tan^{-1}\left(\frac{\epsilon lbow * \sqrt{Pe_x^2 + Pe_y^2}}{Pe_z}\right) \quad (4.63)$$

4.2.3.3 Joint 4

Referring to figure 4.10, joint four is calculated by mapping Pew' and Pew into the $X^3 - Z^3$ plane. The vector Pew' represents the vector from the unrotated primary elbow to the wrist and Pew represents the vector from the rotated primary elbow to the wrist. Joint four is defined by:

$$P^0_{ew} = P_w - D_4 * P_e - P_s \quad (4.64)$$

$$P^3_{ew} = R_0^3 * P^0_{ew} \quad (4.65)$$

$$P^3_{ew'} = \begin{bmatrix} A5 * \cos(q_5) + A6 * \cos(q_5 + q_6) + D7 * \sin(q_5 + q_6) \\ 0 \\ \tilde{D}6 \end{bmatrix} \quad (4.66)$$

$$\cos(q_4) = \frac{P^3_{ew} \odot P^3_{ew'}}{\|P^3_{ew}\| * \|P^3_{ew'}\|} \quad (4.67)$$

$$\sin(q_4) = \text{sgn}(|P^3_{ew} \odot P^3_{ew'}|_y) * \sqrt{1 - \cos^2(q_4)} \quad (4.68)$$

$$q_4 = \tan^{-1}\left(\frac{\sin(q_4)}{\cos(q_4)}\right) \quad (4.69)$$

In summary, the inverse kinematics generates the mapping from the task space to the joint space. Two inverse functions, the Puma-Positioner and the Puma-Platform, were generated to provide mappings from the augmented task space, composed of the redundant task space and Cartesian task space, to the joint space. These mappings produce unique joint vectors through the definition of the pose variables. However, it is not always possible to find a unique solution. There exists certain points within the manipulator's workspace where a unique solution doesn't exist. These points are termed singular points and are the subject of the next chapter.

CHAPTER 5

EFFECT OF INVERSE FUNCTIONS ON MANIPULATOR SINGULARITIES

As a manipulator traverses its workspace, there exists certain points where the manipulator loses some degrees of freedom in its motion. These points are termed singular points. Singular points are generally defined as points in the joint space where the manipulator loses the freedom to travel instantaneously in any direction. This loss of motion is reflected in the Jacobian which becomes rank deficient at singular points. From a different perspective, at a singular point the inverse Jacobian mapping produces infinite joint velocities for bounded task velocities. Singular points can be further characterized by being placed in one of two categories, boundary singularities or internal singularities. Boundary singularities exist on the manipulator's boundaries and define the case when the manipulator is at its maximum reach. In this case, the manipulator is fully extended and no joint motion can cause the end-effector to extend out any further. Internal singularities are singularities internal to the manipulator's workspace.

Singular points are obviously of great importance to the user. They define those areas of the manipulator's workspace where motors could be burnt out due to the generation of large joint speeds and torques. It would be satisfying to be able to generate a workspace that is free of singularities. For non-redundant manipulators, this is impossible. All non-redundant manipulators have singular regions in their workspace and the best that can be done is to locate and avoid them. However, redundant manipulators offer the user some freedom in terms of the singular regions. Redundant manipulators all have singular regions. However, an inverse function can be defined for a redundant manipulator that will have some effect on the singular regions. Redundant manipulators have an infinite number of joint vectors associated with

each task vector. Inverse functions determine a unique joint vector that is associated with each fixed task vector. Since inverse functions determine the joint vectors, they can be used to avoid singular points. However, the tradeoff is that inverse functions generally introduce a third type of singular point. This singular point is termed an algorithmic singularity and indicates the case when the kinematic functions that are defined for the inverse function lose rank. In the next couple of sections, the singular regions of the CIRSSE manipulator will be presented. Also, the effect of the two inverse functions, the Puma-Positioner and the Puma-Platform, on the workspace and singularities of the CIRSSE manipulator will be discussed.

5.1 Characterization of the CIRSSE manipulator's singularities

The CIRSSE manipulator consists of the six dof PUMA atop a three dof platform. In order to evaluate the singularities of this manipulator, the Jacobian was calculated. In order to simplify the form of this Jacobian, the Jacobian matrix of frame seven relative to frame six J_7^6 was calculated. The calculations of these Jacobians are documented in the appendix. The singularities of the CIRSSE manipulator occur at the points where the Jacobian, J_7^6 , is rank deficient. Due to the structure of this manipulator, a necessary condition for it to be in a singular configuration is for the PUMA to be in a singular configuration. These singular configurations of the PUMA are defined as:

- PUMA wrist singularity: This is an internal singular point that occurs when Z^7 and Z^9 are aligned or $q_8 = 0$.
- PUMA elbow singularity: This is a boundary singularity that occurs when the elbow is fully extended or $q_6 = \tan^{-1}(\frac{D_7}{A_6})$.
- PUMA shoulder singularity: This is an internal singular point that occurs when the wrist is aligned with the PUMA's shoulder. This is mathematically

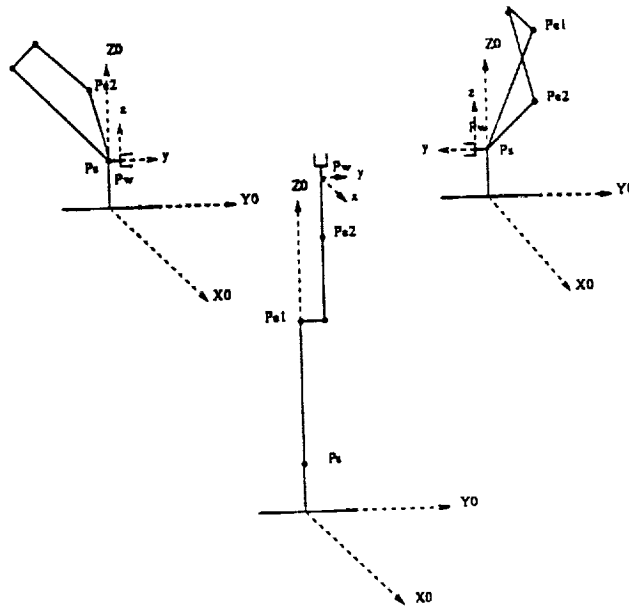


Figure 5.1: Singular configurations of CIRSSE manipulator

defined as when $A5 * \cos(q_5) + A6 * \cos(q_5 + q_6) + D7 * \sin(q_5 + q_6) = 0$.

Figure 5.1 provides sample singular configurations for the CIRSSE manipulator and indicates the possible directions of motion.

5.2 The inverse functions and the workspace singularities

The inverse functions of a manipulator are distinguished from one another by their parameterization of the redundant task space or their kinematic functions. The kinematic functions of the Puma-Positioner are identity functions that simply repeat the first three joints, q_1 , q_2 and q_3 . Based on this parameterization, the Puma-Positioner has no effect on the CIRSSE manipulator's workspace or its singular points. With this parameterization, any possible joint configuration within the reachable joint space can be used. The Puma-Platform has three kinematic functions, $d1$, he and $theta$ that parameterize the redundant task space. This parameterization has two notable effects on the CIRSSE manipulator. First of all,

it defines an unreachable region within the manipulator's workspace based on the PUMA's shoulder offset. Due to joint four being defined by θ_4 in the Puma-Platform, the end-effector is unable to reach inside a circle of radius D_6 that's centered on link three. The second effect of the Puma-Platform is the introduction of two algorithmic singularities. The augmented Jacobian, as defined in the appendix, is composed of the task and redundant task space velocities. The case when the task and redundant task velocities become dependent upon each other and the augmented Jacobian becomes rank deficient is termed an algorithmic singularity. The algorithmic singularities are defined as:

- Primary elbow singularity: The primary elbow is directly above the manipulator's shoulder and no rotation by θ_4 can occur. This is represented by joint three going to zero.
- Secondary elbow singularity: The wrist is close to the primary elbow and θ_4 is at its maximum value. This occurs when $\theta_4 = A_5$ or $q_6 = 267.67$ deg.

It should be noted that the Secondary elbow singularity occurs outside of joint six's joint range. So this singular point is never seen.

In summary, since the inverse functions control the joint configurations for a redundant manipulator, they are capable of altering the manipulator's workspace and singular points. However, one drawback is that the inverse functions tend to introduce algorithmic singular points. The inverse functions, the Puma-Positioner and the Puma-Platform, didn't have a great influence on the manipulator's singularities. The Puma-Platform actually introduced two more algorithmic singularities. However, an inverse function is generally limited to the task its designed for. These two inverse functions were not designed to avoid singular points. Some tasks for which these inverse functions were designed for, such as joint limit avoidance for the Puma-Positioner, will now be discussed.

CHAPTER 6

APPLICATION OF PUMA-POSITIONER TO JOINT LIMIT AVOIDANCE

Robotic manipulators can be viewed as a set of links each interconnected by a joint. Each joint, whether revolute or prismatic, is capable of going through a certain range of motion. The endpoints of this range are termed the joint limits for each joint. Based on these joint limits, the joint space of a manipulator can be divided into a reachable and unreachable subspace. The reachable joint space consists of all the joint vectors whose components reside within their joint limits. Joint limits generally pose a significant problem to the task planner. For instance, given a planned Cartesian path, there is no obvious method to ensure that the corresponding joint path resides within the reachable joint space. If the configuration of the links is not an important issue to the user, then a solution to the joint limit problem presents itself. This solution consists of varying the link configurations for a fixed task vector until a reachable joint vector is produced. For a non-redundant manipulator, there only exists a discrete number of link configurations, termed poses, for a fixed task vector. Due to the existence of only a small number of poses, this solution would not be very effective for a non-redundant manipulator. However, the CIRSSE manipulator is a redundant manipulator with three extra dofs. By using the redundant task space and the pose variables, an infinite number of link configurations can be generated for a fixed task vector. A solution to the joint limit problem for the CIRSSE manipulator is hereby presented that augments the Puma-Positioner inverse function with a decision algorithm. This algorithm generates the redundant task vector, q_1 , q_2 and q_3 based upon certain predefined criteria. This criteria is enumerated as follows:

- The task vector constraints, position and orientation of the end-effector, must be met. The position constraints have the higher priority and are the first to be met.
- The resultant joint vector must reside in the reachable joint space. There are some cases in which the resultant joint vector is in the unreachable joint space. In this case, a solution is found and termed close that consists of clipping the joints at their joint limits.
- The final criteria is to minimize flipping the PUMA's pose during path traversal. Every time the PUMA's pose is flipped, the resultant joint vector deviates significantly from the previous joint values. This doesn't allow for a smooth traversal of a given Cartesian path. The PUMA's pose is flipped only when it's the only solution to meeting the previous constraints.

To summarize, the joint limit avoidance routine can be broken down into a series of steps. Given the task vector, three user defined control variables and the PUMA's pose, the decision algorithm generates a set of Puma-Positioner configuration variables and the PUMA's pose variables. These variables serve to satisfy the previously defined criteria. These pose and configuration variables are then used in conjunction with the task vector by the Puma-Positioner to generate the corresponding joint vector. A thorough description of the decision algorithm will now be given.

6.1 Definition of the decision algorithm

The decision algorithm determines the values of the redundant task vector, (q_1, q_2, q_3) and the PUMA's pose, *right_left, wrist* and *wst_to_e2*. In order to produce these parameters, some simplifications were made. The manipulator's link configurations were restricted to those displayed in figures 6.3-6.4. Also, joint four of

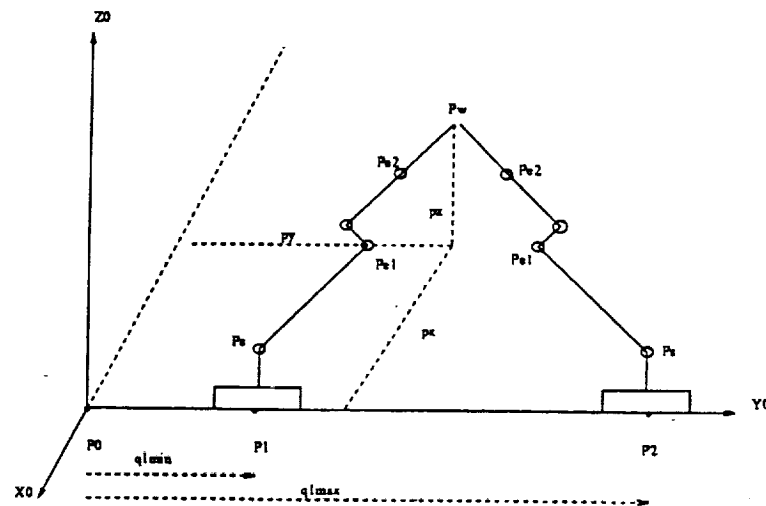


Figure 6.1: Generating configuration variable q_1

the manipulator was fixed at zero to ensure that the links were always in the plane perpendicular to the $X^0 - Y^0$ plane. This is explained in greater detail in the section on meeting the orientation constraints. The configuration and pose variables are now defined to meet the predefined constraints.

6.1.1 Configuration variable q_1 (Joint 1)

Joint one is a linear joint and corresponds to a translation of the entire manipulator along the linear rails. Since this joint moves the entire manipulator, its motion is restricted and controlled by the user. There are two cases for which joint one motion is generated. Movement is initiated if the desired position is outside the manipulator's reach or if the desired position is less than some user defined threshold. If the desired position is outside the manipulator's reach and motion along the linear rails would place the manipulator within reach of the desired position, this motion is taken. Referring to figure 6.1, there exists a range of values for q_1 that would allow the manipulator to reach the target. The two positions, labeled $P1$ and $P2$, displayed in this figure indicate the end-points of this range. The configuration

variable q_1 is defined by:

$$q_1 = q_1^{old} + py - \text{sgn}(py) * 2 * (0.5 - \text{ctrl1}) * \sqrt{rmax^2 - (px^2 + pz^2)} \quad (6.1)$$

where $rmax$ is defined in figure 6.2 and table 6.1 and ctrl1 is a user defined variable that varies between zero and one. This variable determines how far the platform will move in order to reach a point outside its current workspace. A value of zero defines the smallest distance that the platform can move and still reach the target. If the target is outside the manipulator's reach even with motion along the linear rails, then the manipulator is brought as close as possible to the target. This point is defined as having the manipulator being directly in front of the target or at the closest joint one limit. In the second case, a minimum threshold is determined by a second control variable, ctrl2 . This variable defines the threshold p_{thr} by:

$$p_{thr} = pmin + \text{ctrl2} * (pmax - pmin) \quad (6.2)$$

where $pmin = x2m - r1$ and $pmax = rmax$ are manipulator defined values that are defined in figure 6.2 and table 6.1. Joint one is then defined as

$$q_1 = q_1^{old} + py + \sqrt{p_{thr}^2 - (px^2 + pz^2)} \quad (6.3)$$

6.1.2 Configuration variable q_2 (Joint 2)

Joint two places the end-effector in the $X^0 - Y^0$ plane. As in (4.38), joint two is defined by:

$$q_2 = \tan^{-1} \left(\frac{-px * \bar{D}6 + \text{right_left} * py * \sqrt{px^2 + py^2 - \bar{D}6^2}}{py * \bar{D}6 + \text{right_left} * px * \sqrt{px^2 + py^2 - \bar{D}6^2}} \right) \quad (6.4)$$

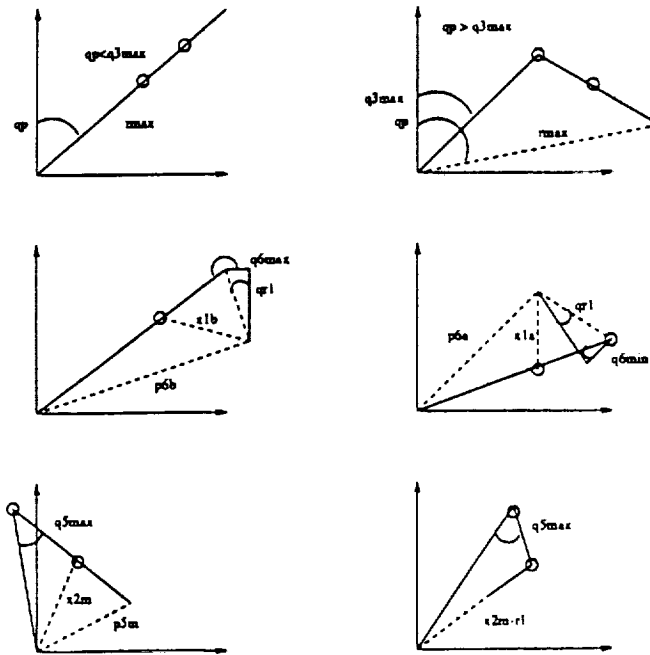


Figure 6.2: Joint limit avoidance: Distance constraints

rmax	$\sqrt{D4^2 + (A5 + R1)^2 - 2 * D4 * (A5 + R1) * \cos(q3max - qp)}$
x1b	$\sqrt{A5^2 + R1^2 - 2 * A5 * R1 * \cos(q6max + qr1 - \pi)}$
p6b	$\sqrt{(D4 + A5)^2 + R1^2 - 2 * (D4 + A5) * R1 * \cos(q6max + qr1 - \pi)}$
x1a	$\sqrt{A5^2 + R1^2 - 2 * A5 * R1 * \cos(q6min + qr1)}$
p6b	$\sqrt{(D4 + A5)^2 + R1^2 - 2 * (D4 + A5) * R1 * \cos(q6min + qr1)}$
x2m	$\sqrt{D4^2 + A5^2 - 2 * D4 * A5 * \cos(q5max)}$
p5m	$\sqrt{D4^2 + (A5 + R1)^2 - 2 * D4 * (A5 + R1) * \cos(q5max)}$

Table 6.1: Distance constraints

If this calculated value is outside joint two's limits then the *right_left* pose is flipped and joint two is recalculated. Due to the definition of the *right_left* pose, joint two can effectively cover the entire $X^0 - Y^0$ plane by flipping this pose whenever joint two is outside its limits. Joint two actually covers a donut centered on the $X^0 - Y^0$ plane. The outer radius is described by *rmax* while the inner radius is defined by the length of the PUMA's shoulder offset, \bar{D}_6 .

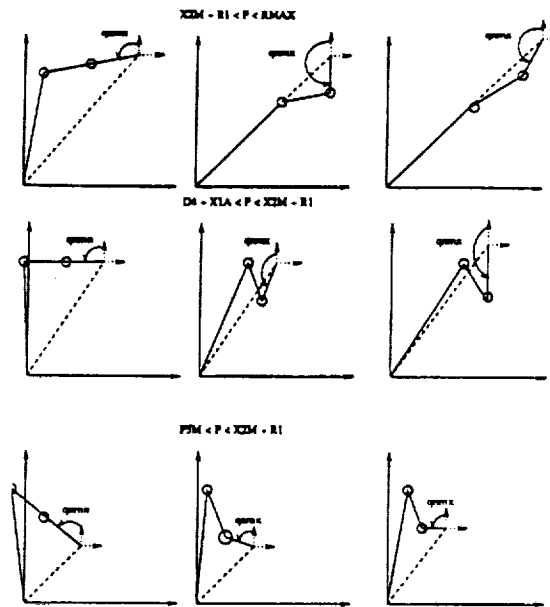


Figure 6.3: Range of q_n angles for wrist ABOVE secondary elbow

6.1.3 Configuration variable q_3 (Joint 3)

Joint three and the *wst_to_e2* pose are used to meet the vertical position constraint and the orientation constraints. Initially, the current *wst_to_e2* pose is used to determine the range of joint three angles that would match the position constraints. If no range is found then the *wst_to_e2* pose is switched and a new range of joint three angles is found. If the orientation constraints fall within the joint three range, a joint three value is selected through the user defined control variable, *ctrl3*. However, if the orientation constraints fall outside the joint three range and the *wst_to_e2* pose hasn't already been switched, then the *wst_to_e2* pose is switched. If after switching the *wst_to_e2* pose, the position and orientation constraints still can't be met, then the joints are clipped at their limits and the joint vector is returned with the appropriate status. A detailed description will now be given of how the joint three range is calculated.

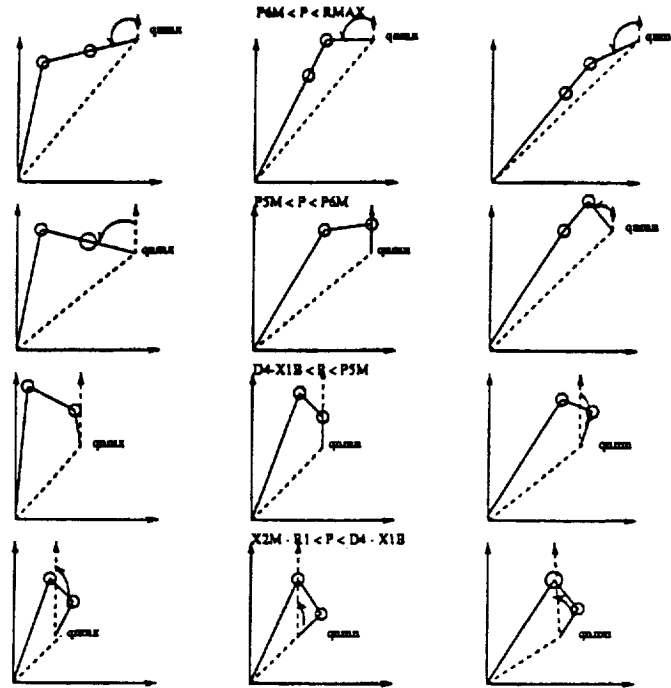


Figure 6.4: Range of q_n angles for wrist BELOW secondary elbow

6.1.3.1 Meeting the position constraints

Figure 6.2 describes standard link configurations of the last three links with joint four set to zero and the PU^{MA}'s shoulder offset parallel to the $X^0 - Y^0$ plane. Since, as seen in this figure, the angle q_n relates directly to the end-effector's orientation, the angles that meet the distance constraints are expressed in q_n instead of in q_3 . The position constraints can be expanded into a set of distance constraints that are a factor of the link lengths and the joint limits. These distance constraints are described in figure 6.2 and tabulated in table 6.1. These constraints and the *wst_to_e2* pose describe the range of q_n angles generated.

Figure 6.3 describes the configurations that the links are allowed to be in for the case where the wrist is above the secondary elbow. These configurations are placed in groups based on the distance p of the wrist from the shoulder. Each group indicates a range of q_n angles where q_n varies from q_{nmn} to $q_{nm\alpha}$. Based upon

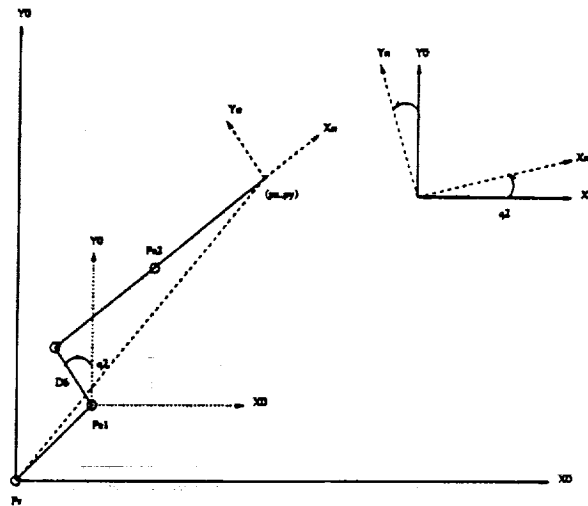


Figure 6.5: Description of orientation coordinate frame

the distance p , a range of qn angles is generated. It should be noted that if p is less than $p5m$, no range of qn angles can be generated. This means that for p less than $p5m$, the wrist can't be placed above the secondary elbow.

Figure 6.4 describes the configurations that the links are allowed to be in for the case where the wrist is below the secondary elbow. As in the previous *wst_to_e2* pose, the range of qn angles generated is based on the distance p . This *wst_to_e2* pose is more flexible than the previous pose. If a position can't be reached in this pose then a clipped joint solution is generated. As seen in figure 6.4, such a case occurs when p is less than $x2m - r1$.

6.1.3.2 Meeting the orientation constraints

Once a range of qn angles is generated that meets the position constraints, the orientation constraints are then applied to this range. The orientation of this manipulator is defined by the last three joints of the manipulator, $q7$, $q8$ and $q9$. The orientation constraints are based upon the joint limits of these joints. Joint nine has a joint range that exceeds 360 degrees. This indicates that a solution can

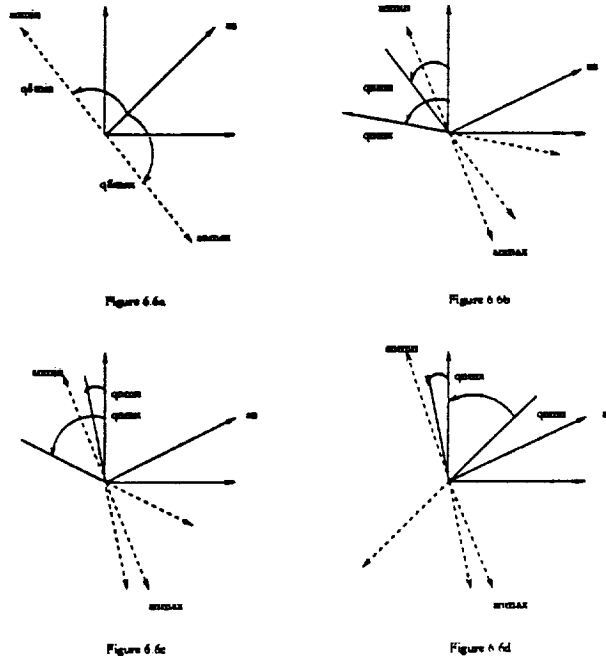


Figure 6.6: Description of orientation constraints

always be found that resides within the joint limits of joint nine. Joint seven has a joint range that's less than 360 degrees. However, by switching the *wrist* pose whenever joint seven is out of range, a solution can still be generated that's within joint seven's joint range. However, if the *wrist* pose is flipped, this means that any objects in the end-effector's grasp will be flipped also. A status flag is set to alert the user when this occurs. The orientation constraints are thus based on joint eighth's joint limits. The approach vector a is determined relative to the coordinate system defined in figure 6.5. It should be noted that joint four was set to zero to ensure that the $X^n - Y^n$ plane was always parallel to the $X^0 - Y^0$ plane. Once the vector a^n is calculated then, as displayed in figure 6.6a, two vectors, $amin^n$ and $amax^n$ are calculated based on joint eighth's limit. These vectors determine the section of the $X^n - Z^n$ that the end-effector has to be in so that the joint solution generated is within joint eighth's limits. Three cases are displayed in figures 6.6b-6.6d. In the first case (Figure 6.6b), the entire q^n range meets the orientation constraints.

In the second case (Figure 6.6c), one endpoint of the qn range doesn't meet the constraints. In this case, a new qn endpoint is defined as the closest orientation constraint. The last case (Figure 6.6d) shows the entire qn range not meeting the orientation constraints. In this case, the qn range is reduced to the endpoint that's closest to an orientation constraint and a status flag is set. Once a range of qn angles is generated, one of them is selected through

$$qn = qnmn + ctrl3 * (qnm x - qnmn) \quad (6.5)$$

where $ctrl3$ is a user defined control variable that varies between zero and one. Given the configuration variables, q_1 , q_2 and q_3 , the PUMA's pose and the task vector, the Puma-Positioner algorithm is used to calculate the corresponding joint vector.

In Summary, the joint limit avoidance scheme is defined by augmenting the Puma-Positioner with a decision algorithm. This decision algorithm determines the configuration variables and the PUMA's pose based upon certain predefined criteria. The Puma-Positioner then uses the configuration variables, the PUMA's pose and the task vector to generate a reachable joint vector. This algorithm is typical of the approach taken towards inverse functions in this thesis. The algorithm is not generic and can't be easily applied to all manipulators. However, due to its specialization, it's both fast and efficient. These are traits that are especially appreciated in the obstacle avoidance scheme that will now be discussed.

CHAPTER 7

APPLICATION OF THE PUMA-PLATFORM TO OBSTACLE AVOIDANCE

The CIRSSE robotic facilities consists of two nine dof manipulators operating along a linear rail. Each manipulator operates in an environment that's filled with potential obstacles. Collision avoidance of the end-effector can be implemented by careful planning and is generally a task of the task planner. However, since these manipulators are redundant, the position of the end-effector doesn't give adequate information about the location of the links. This means that although the end-effector's path can be planned so that it doesn't collide with any obstacles, there is no guarantee that the links will not collide with some obstacle. A collision avoidance scheme is desired that would ensure that the internal links of the manipulator don't collide with each other or other obstacles in the workspace. Two criteria that are necessary for the collision avoidance scheme are that its fast and that it operates in the null-space of the end-effector. The first criteria is needed to ensure that the obstacle avoidance scheme can be implemented on-line and can respond fast enough to an obstacle entering the manipulator's workspace. The second criteria is necessary so that the generated internal link motion doesn't affect the the end-effector and cause it to deviate from its path. The collision avoidance scheme that is presented in this thesis is based on the Puma-Platform inverse function. The redundant task space or secondary task space parameterizes the internal link motion through the configuration variables, $d1$, he and $theta$. This parameterization resolves the link configurations into three disjoint sets that are each controlled by one of the configuration variables. Obstacles in the workspace induce motion in one of these sets and thus cause the appropriate configuration variable to be altered. The basic algorithm can be dissolved into a series of steps which will now be defined.

OBSTACLE AVOIDANCE ALGORITHM:

- STEP 1: The current joint angles are used to calculate the manipulator's present redundant configuration. A coordinate system is defined by the calculated configuration parameters and attached to the manipulator's links. A set of distances d_{ik} are then calculated relative to this coordinate system. The distance d_{ik} represents the distance from link i to obstacle k .
- STEP 2: Each link has a radially decaying force field surrounding it. The intersection of an obstacle with this field determines the amount of force being exerted by this obstacle on the link. This force is used to define the desired increase in d_{ik} , δd_{ik} . The desired distance $d_{ik} = d_{ik} + \delta d_{ik}$ is mapped into the appropriate configuration variables.
- STEP 3: Once the new configuration variables are found, the Puma-Platform inverse function is used to determine the corresponding joint angles. The manipulator's links are moved to this new configuration and the process repeats itself.

This obstacle avoidance scheme is a process that relies solely on information obtained at each time k . No past information or estimated future information is used. At each time k , a snapshot of the manipulator and its workspace is taken. Based on the information contained in this snapshot, the links are reconfigured to avoid any obstacles. In the following sections, the various components of the obstacle avoidance algorithm will be discussed. A description of the modeling process will be given. The generation of the closest point of contact between the models will be covered. Also, the mathematically defined force fields assigned to each link will be discussed. Finally, a description of the mapping from the distances d_{ik} to the configuration variables will be given.

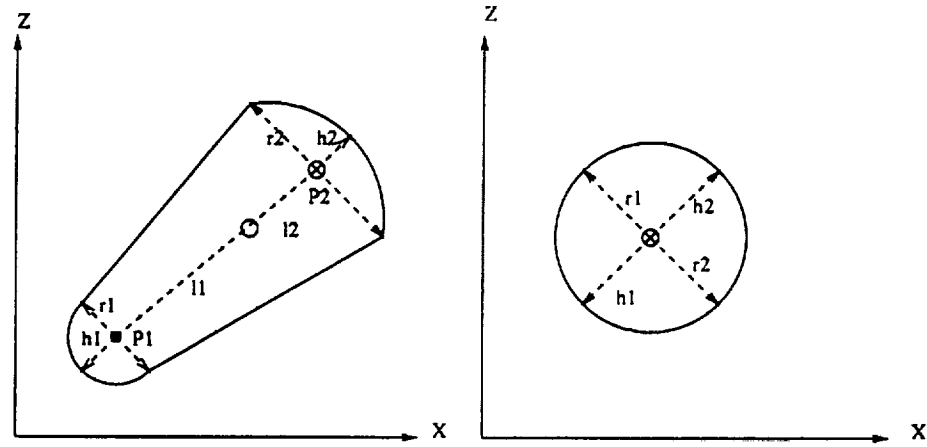


Figure 7.1: Link and Obstacle model

7.1 Link and Obstacle modeling

Both the links and the obstacles are modeled with a simple convex structure. This structure has the form of a cylinder with varying radii where each end is capped by a variable height cone. The data structure that defines this structure takes the form of

$$struct = \left[P1 \ h1 \ r1 \ l1 \ P2 \ h2 \ r2 \ l2 \right] \quad (7.1)$$

where P_i represents the position of one end of the cylinder, r_i is the radius of the cylinder at that end, h_i the height of the cone attached at that end and l_i is the distance from the center of the cylinder to its end. A description of the different forms this structure can take is given in figure 7.1

7.2 Generating the closest point of contact

The last three links and the link offset are all mapped into a plane that passes through the position vector P_{sw} and the primary elbow P_{el} . This plane rotates

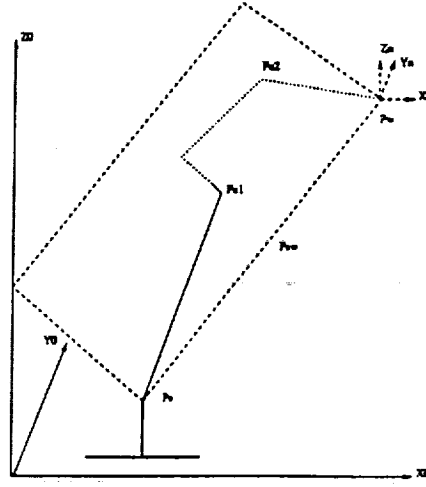


Figure 7.2: Description of rotation matrix $R(n, \theta)$

around the vector Psw as the primary elbow Psl is rotated by θ around Psw . Figure 7.2 shows this plane and indicates the rotational matrix, $R(n, \theta)$, that describes to this plane. This matrix is described by:

$$R(n, \theta) = \begin{bmatrix} n_x^2 * V + C & n_x * n_y * V - n_z * S & n_x * n_z * V + n_y * S \\ n_x * n_y * V + n_z * S & n_y^2 * V + C & n_y * n_z * V - n_x * S \\ n_x * n_z * V - n_y * S & n_x * n_z * V + n_y * S & n_z^2 * V + C \end{bmatrix}$$

$$C = \cos(\theta)$$

$$S = \sin(\theta)$$

$$V = 1 - \cos(\theta)$$

$$n = \frac{Psw}{\|Psw\|} \quad (7.2)$$

The rotational matrix $R(n, \theta)$ describes the coordinate frame $(X^n - Y^n - Z^n)$ that, as described in figure 7.2, is attached to the vertical plane. This coordinate frame parameterizes the link motion into the motion that occurs in the plane and the motion that rotates the plane around Psw . Each obstacle's position is found

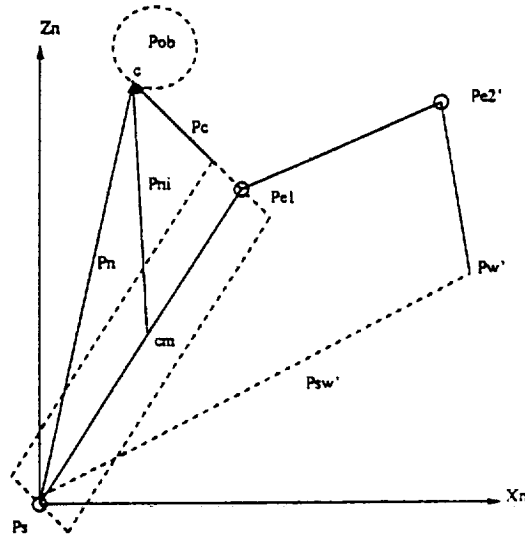


Figure 7.3: Generation of model based closest point of contact

relative to this coordinate frame and determines the obstacle's influence on the two parameterized link motions. Figure 7.3 gives an example of an obstacle relative to the coordinate frame attached to the links. The links and obstacles are all modeled with the previously defined structure. Referring to figure 7.3, it can be seen that three pieces of information are extracted from each link and obstacle pair. This information consists of three vectors, Pc , Pni and Pn . These vectors are defined as follows:

- Pc : The closest distance between the link and the obstacle.
- Pni : The vector from the center of mass of the link, labeled cm , to the closest point on the obstacle, labeled c .
- Pn : The distance from the manipulator's shoulder Ps to the closest point on the obstacle.

These three vectors, Pc , Pni and Pn are used by the obstacle avoidance algorithm to generate the configuration variables. The code that generates these vectors

is kept separate from the main algorithm. This ensures program modularity and allows the user to change the model description.

7.3 Generation of force fields

Each link has a mathematically defined decaying field assigned to it. The point of intersection of the obstacle with a link's field determines how much force is being applied to the link. This force determines the change in distance, δd_{ik} that link i will undergo to ensure that the new distance of link i to obstacle k is equal to $d_{ik} = d_{ik} + \delta d_{ik}$. The change in distance is defined by:

$$K_0 = \frac{1}{1 + H_i * |d'_{ik}|} \quad (7.3)$$

$$\delta d_{ik} = \frac{K_0 * F_i}{1 + W_i * |d_{ik}|} \quad (7.4)$$

where H_i and W_i are link defined constants, F_i represents the maximum allowed δd_{ik} for link i and d_{ik} and d'_{ik} represent two perpendicular components of the vector from link i to obstacle k .

7.4 Assignment of the Configuration variables

The redundant task space is subdivided into three separate subspaces, each described by one of the configuration variables, $d1$, he or $theta$. Each one of the configuration variables describes a distinct type of internal link motion. The configuration variable $d1$ describes the motion corresponding to a fixed end-effector with the platform translating along the linear rails. The variable he describes the motion of links three, four and five in the vertical plane that passes through the position vector Psu and the primary elbow $Pe1$, as described in figure 7.2. Finally, the variable $theta$ describes the motion of the plane containing the last three links as

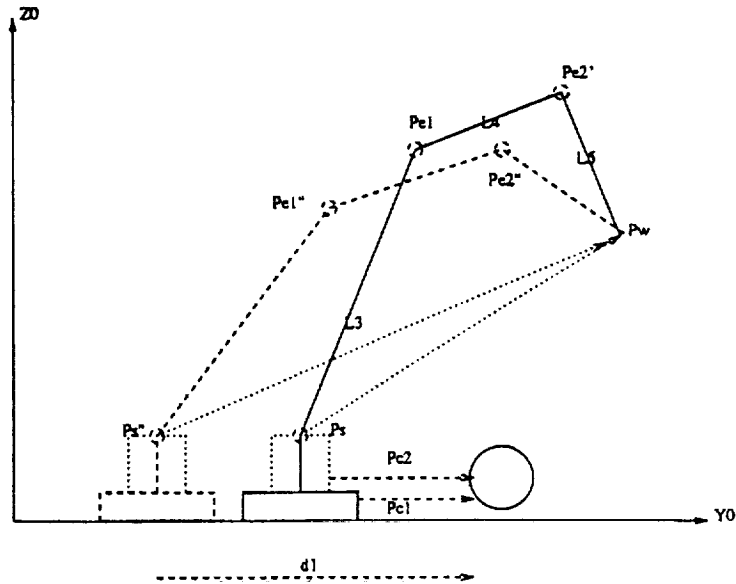


Figure 7.4: Obstacle Avoidance: Definition of $d1$

that plane rotates around the position vector Psw . This is also described in figure 7.2. The assignment of each configuration variable will now be described.

7.4.1 Assignment of $d1$

The configuration variable $d1$ deals with the translation of the platform's base. Movement of the platform is only initiated when an obstacle comes into contact with the force field surrounding links one and two, as described in figure 7.4. The assignment of $d1_{ik}$ then becomes

$$d1_{ik} = d1_{old} - \text{sgn}(d_{ik}) * \delta d_{ik} \quad (7.5)$$

where d_{ik} represents the component of the vector from link i to obstacle k that's parallel to Y^0 and $d1_{ik}$ represents the response of links one and two to obstacle k .

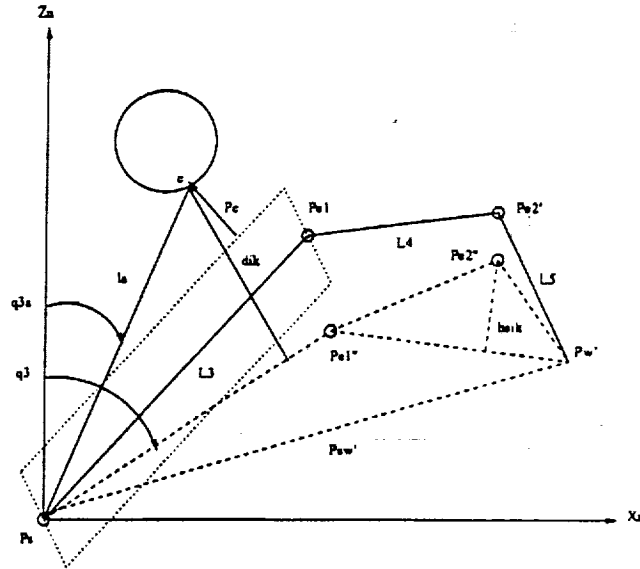


Figure 7.5: Obstacle Avoidance: Definition of h_e (Link 3)

7.4.2 Assignment of h_e

The configuration variable h_e deals with the motion of the links in the perpendicular plane described in figure 7.2. Motion is initiated through the variable h_e whenever an obstacle comes into contact with the force field surrounding links three, four or five. The magnitude of this motion is determined by the distance from the obstacles to the links relative to the coordinate frame, $(X^n - Y^n - Z^n)$. Each of the links generates a value for h_e based on the effect of the obstacle on that link. These values for h_e are later resolved into one value in the conflict resolution algorithm.

7.4.2.1 Link $i = 3$

Referring to figure 7.5, $h_{e_{ik}}$ is defined by:

$$d_{ik} = d_{ik} + \delta d_{ik} \quad (7.6)$$

$$l_a = \sqrt{P_{n_x}^2 + P_{n_z}^2} \quad (7.7)$$

$$q_{3a} = \tan^{-1}\left(\frac{P_{n_x}}{P_{n_z}}\right) \quad (7.8)$$

$$rb = \sqrt{Pn_y^{i2} + (0.5 * A5 - Pn_z^i)^2} \quad (7.14)$$

If ra is less than rb then he_{4k} is calculated by:

$$ra_{ik} = ra + \delta d_{ik} \quad (7.15)$$

$$la = \sqrt{Pn_x^{i2} + Pn_z^{i2}} \quad (7.16)$$

$$q3a = \tan^{-1}\left(\frac{Pn_x^i}{Pn_z^i}\right) \quad (7.17)$$

$$\cos(qa) = \frac{la^2 + D4^2 - ra_{ik}^2}{2 * la * D4} \quad (7.18)$$

$$qa = \tan^{-1}\left(\frac{\sqrt{1 - \cos^2(qa)}}{\cos(qa)}\right) \quad (7.19)$$

$$q3 = q3a + \text{sgn}(q3p - q3a) * qa \quad (7.20)$$

$$Pe = \begin{bmatrix} \sin(q3) \\ 0 \\ \cos(q3) \end{bmatrix} \quad (7.21)$$

$$x1 = \|P_{sw}' - Pe\| \quad (7.22)$$

$$he_{4k} = \sqrt{\frac{(2 * A5 * x1)^2 - (x1^2 + A5^2 - r1^2)^2}{4 * x1^2}} \quad (7.23)$$

otherwise he_{4k} is calculated by:

$$rb_{ik} = rb + \delta d_{ik} \quad (7.24)$$

$$Lb = Pn^i - P_{sw}' \quad (7.25)$$

$$lb = \|Lb\| \quad (7.26)$$

$$qnb = \tan^{-1}\left(\frac{-Lb_x}{Lb_z}\right) \quad (7.27)$$

$$\cos(qb) = \frac{lb^2 + r1^2 - rb_{ik}^2}{2 * lb * r1} \quad (7.28)$$

$$qb = \tan^{-1}\left(\frac{\sqrt{1 - \cos^2(qb)}}{\cos(qb)}\right) \quad (7.29)$$

$$q3 = q3b + \text{sgn}(q3p - q3b) * qb \quad (7.30)$$

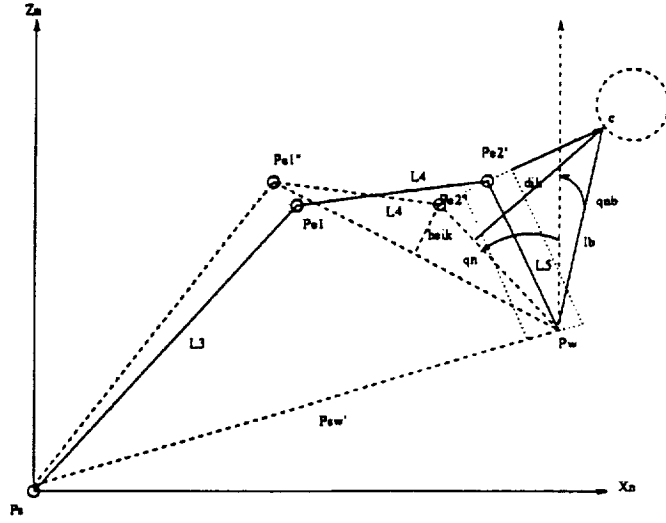


Figure 7.7: Obstacle Avoidance: Definition of h_e (Link 5)

$$P_e = \begin{bmatrix} \sin(q3) \\ 0 \\ \cos(q3) \end{bmatrix} \quad (7.31)$$

$$x1 = \|P_{sw'} - P_e\| \quad (7.32)$$

$$h_{e_{4k}} = \sqrt{\frac{(2 * A5 * x1)^2 - (x1^2 + A5^2 - r1^2)^2}{4 * x1^2}} \quad (7.33)$$

where $h_{e_{4k}}$ represents the response of link 4 obstacle k.

7.4.2.3 Link 5

Referring to figure 7.7, $h_{e_{5k}}$ is defined by:

$$d_{ik} = d_{ik} + \delta d_{ik} \quad (7.34)$$

$$Lb = P_{n'} - P_{sw'} \quad (7.35)$$

$$lb = \|Lb\| \quad (7.36)$$

$$qnb = \tan^{-1}\left(\frac{-Lb(1)}{Lb(3)}\right) \quad (7.37)$$

$$d_{ik} = d_{ik} + \delta d_{ik} \quad (7.42)$$

$$la = \sqrt{Pn_x^i{}^2 + Pn_z^i{}^2} \quad (7.43)$$

$$q3a = \tan^{-1}\left(\frac{Pn_x^i}{Pn_z^i}\right) \quad (7.44)$$

$$Py = la * \sin(q3a - qp) \quad (7.45)$$

$$r = \sqrt{|Pni_x^i|^2 + Py^2} \quad (7.46)$$

$$alf = \tan^{-1}\left(\frac{d_{ik}}{\sqrt{r^2 - d_{ik}^2}}\right) \quad (7.47)$$

$$bet = \tan^{-1}\left(\frac{|Pni_x^i|}{Py}\right) \quad (7.48)$$

$$theta_{ik} = theta_{old} + \text{sgn}(Pni(1, i) * (alf - bet)) \quad (7.49)$$

where $theta_{ik}$ rotates the perpendicular plane the desired distance d_{ik} away from the obstacles.

7.5 Resolving obstacle conflicts

The group of configuration variables ($d_{ik}, he_{ik}, theta_{ik}$) indicate the individual responses of each link to each obstacle. This group is resolved into one set of configuration variables ($d, he, theta$) by a conflict resolution algorithm. This algorithm utilizes a simplistic approach by choosing from the group of configuration variables the ones that move the links the furthest away from the obstacles. If two opposing values are generated for one configuration value, the midpoint between these values is chosen as the magnitude of the configuration variable. Once the configuration variables are obtained, the final step of the algorithm is implemented and the corresponding joint vector is generated.

In summary, the obstacle avoidance scheme relies only on the information obtained at each time k . This information, which consists of the link and obstacle locations, is used to generate the new configuration for the links to be placed in.

The resultant link motion is generated in the null-space of the end-effector and serves to maximize the distance between the links and the obstacles. This obstacle avoidance algorithm, as is the case for the joint limit avoidance algorithm, is not a generic algorithm. It can't be easily applied to other manipulators. However, this specialization lends it speed and efficiency. The results from testing and simulating the obstacle avoidance and joint limit avoidance algorithms and the inverse functions will now be presented.

CHAPTER 8

TEST AND SIMULATION RESULTS

The results produced by this thesis can be divided into two categories, the test results for the inverse functions and the simulation results for the joint limit avoidance and obstacle avoidance routines.

The inverse functions, the Puma-Positioner and the Puma-Platform, were both implemented in C. Since these inverse functions generated one-to-one mappings between the joint space and augmented task space of the CIRSSE manipulator, the following testing procedure was used.

TESTING PROCEDURE:

- The manipulator's joint space was sampled and an appropriate joint vector was selected.
- The augmented task vector was calculated as a function of this joint vector. The augmented task vector is comprised of the Cartesian task vector and the redundant task vector.
- The joint vector was used to calculate the manipulator's pose. This pose and the augmented task vector was then used by the inverse function to generate the corresponding joint vector.
- This calculated joint vector was compared with the sampled joint vector through the use of a suitable tolerance.

Table 8.1 summarizes the results of running this testing procedure on the two inverse functions. The joint range category of the table indicates whether the entire joint range was sampled or a specific range around a singularity was sampled. It should be noted that around the singular regions the tolerance level had to be

Inverse functions	Joint Range	Iterations	Errors	Tolerance (deg)
Puma-Positioner	all	3.2×10^6	4100	0.005
Puma-Positioner	q_3	10^5	120	0.5
Puma-Positioner	q_6	10^5	120	0.5
Puma-Positioner	q_8	10^5	120	0.5
Puma-Platform	all	3.2×10^6	4000	0.005
Puma-Platform	q_3	10^5	100	0.5
Puma-Platform	q_6	10^5	100	0.5
Puma-Platform	q_8	10^5	100	0.5

Table 8.1: Test results for the inverse functions

increased. This was done because the singular points of the inverse functions tend to exhibit numerical instabilities.

Simulations of the joint limit avoidance and obstacle avoidance routines were generated for the CIRSSE manipulator through the use of SILMA Cimstation package. The joint limit avoidance routine was implemented in C. The simulations of the joint limit avoidance routine are displayed in figures 8.1-8.4. Figures 8.1-8.2 display the different possible configurations for a fixed task vector with the PUMA's elbow up. Figures 8.3-8.4 display a different set of link configurations for the same task vector with the PUMA's elbow down. From these figures it should be clear that there exists many different configurations for a fixed task. The joint limit avoidance scheme selects those that generate reachable joint vectors and then allows the user to choose a link configuration from this set. The obstacle avoidance scheme was implemented in matlab code. Figures 8.5-8.8 display a sample run of two manipulators performing two tasks simultaneously. These figures describe the case where the right manipulator is running the obstacle avoidance routine while performing a task. The left manipulator is also performing a task that takes it into the workspace of the right manipulator. The obstacle avoidance routine on the right manipulator causes the right manipulator's links to move away from the left manipulator without deviating from its task. This simulation displays the power of the obstacle avoidance

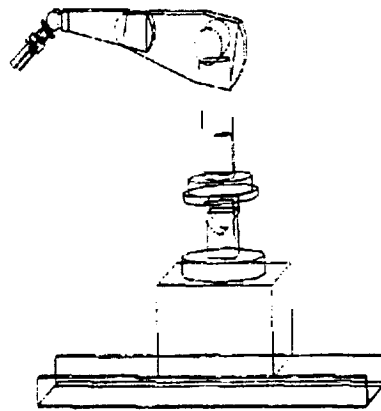


Figure 8.1: Joint-Limit-Avoidance: Wrist BELOW Secondary elbow

routine. It allows the two manipulators to perform tasks that takes them into each others workspace with their links being able to reconfigure themselves to avoid each other.

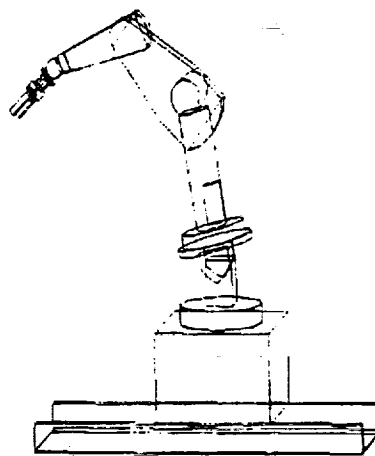


Figure 8.2: Joint-Limit-Avoidance: Wrist BELOW Secondary elbow

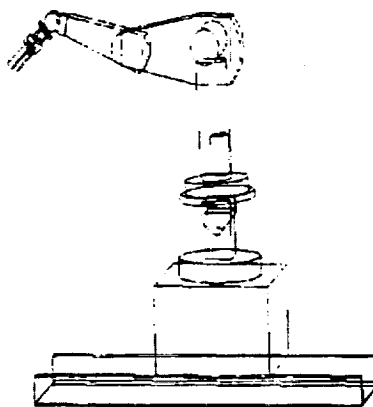


Figure 8.3: Joint-Limit-Avoidance: Wrist ABOVE Secondary elbow

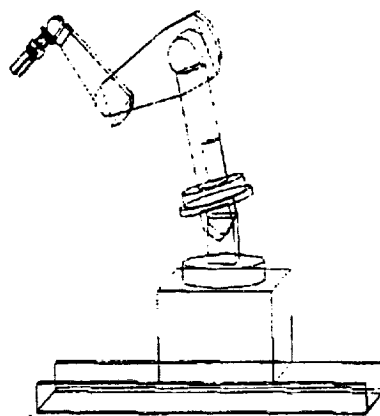


Figure 8.4: Joint-Limit-Avoidance: Wrist ABOVE Secondary elbow

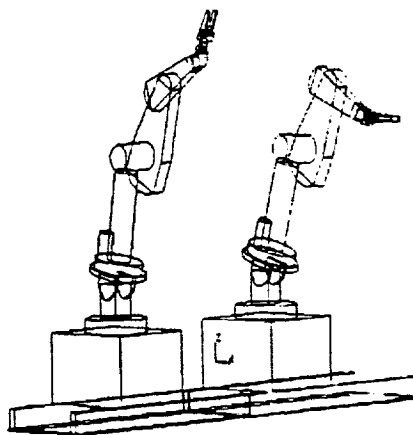


Figure 8.5: Obstacle-Avoidance: Manipulators at time $t = 0.0$

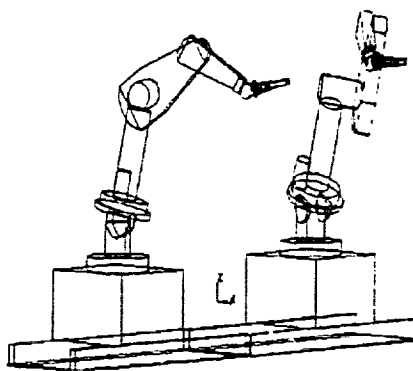


Figure 8.6: Obstacle-Avoidance: Manipulators at time $t = 1.0$

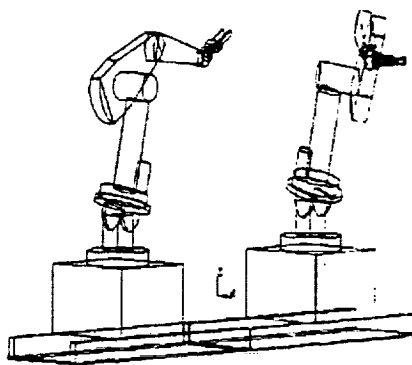


Figure 8.7: Obstacle-Avoidance: Manipulators at time $t = 2.0$

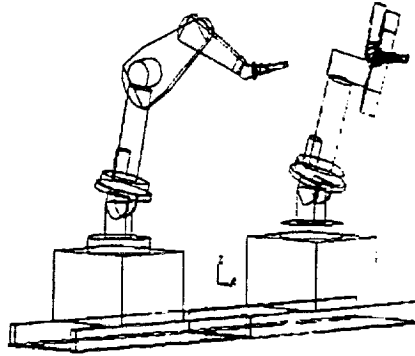


Figure 8.8: Obstacle-Avoidance: Manipulators at time $t = 3.0$

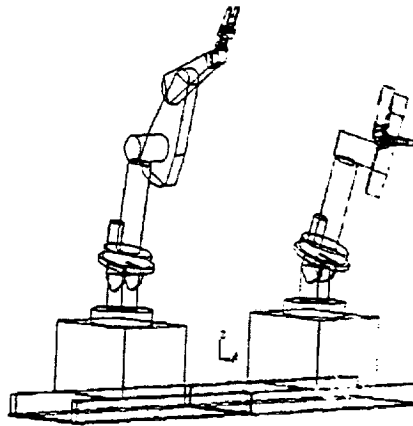


Figure 8.9: Obstacle-Avoidance: Manipulators at time $t = 4.0$

CHAPTER 9

CONCLUSION

This thesis has resolved the redundancy resolution problem of the CIRSSE redundant manipulator through the development of two inverse functions. These inverse functions, the Puma-Positioner and the Puma-Platform, each augmented the Cartesian space with a redundant task space which served to provide the necessary extra dofs to the task space. The redundant task space or secondary task space of each inverse function was designed with a specific task in mind. The secondary task space of the Puma-Positioner allowed the user to position the base of the PUMA. The secondary task space of the Puma-Platform enabled the user to place and configure the end-effector and the links. These inverse functions were then applied to the areas of joint limit and obstacle avoidance. The Puma-Positioner was incorporated into a joint limit avoidance scheme by defining its secondary task vector based on link and joint constraints. The Puma-Platform's secondary task vector was defined by the location of obstacles in the manipulator's workspace. These obstacles caused the links to reconfigure themselves without affecting the end-effector's motion.

9.1 Future Work

The inverse functions in this thesis were developed specifically for use in the joint limit and obstacle avoidance routines. Other avenues of research include developing inverse functions that deal with manipulator singularities and manipulability. Also, these inverse functions only provide a kinematic structure for the manipulators. In order to effectively control these manipulators, a dynamic approach to redundant manipulators must be developed. Such an approach could be developed by combining control theory and redundancy resolution methods.

APPENDIX A

Symbol Definitions

- $a \otimes b$: The cross product of the vectors a and b .
- $a \odot b$: The dot product of the vectors a and b .
- $\alpha * \beta$: The scalar multiplication of the scalars α and β .

APPENDIX B

Modified Denavit-Hartenberg Representation

The homogeneous transformation relating the i^{th} frame to the $(i-1)^{th}$ frame, as given in Craig[16], is described by

$$T_i^{i-1} = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_{i-1} \\ S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S\alpha_{i-1} & -d_i S\alpha_{i-1} \\ S\theta_i S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{i-1} & d_i C\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (B.1)$$

where $(a_i, \alpha_i, d_i, \theta_i)$ are defined as:

- a_i : the distance from Z_i to Z_{i+1} measured along X_i
- α_i : the angle between Z_i to Z_{i+1} measured about X_i
- d_i : the distance from X_{i-1} to X_i measured along Z_i
- θ_i : the angle between X_{i-1} to X_i measured about Z_i

APPENDIX C

The Jacobian of the CIRSSE manipulator

The Jacobian is represented by

$$J_e^0 = \begin{bmatrix} 0 & Z_2^0 & Z_3^0 & Z_4^0 & Z_5^0 & Z_6^0 & Z_7^0 & Z_8^0 \\ Z_1^0 & Z_2^0 \times P_{2,e}^0 & Z_3^0 \times P_{3,e}^0 & Z_4^0 \times P_{4,e}^0 & Z_5^0 \times P_{5,e}^0 & Z_6^0 \times P_{6,e}^0 & Z_7^0 \times P_{7,e}^0 & Z_8^0 \times P_{8,e}^0 \end{bmatrix} \quad (C.1)$$

In order to evaluate the manipulator's singularities, the Jacobian of frame seven relative to frame six, J_7^6 , as described by Fijany and Bejczy[17], was calculated and is described by:

$$J_7^6 = \begin{bmatrix} 0 & j_{12} & S_4 C_{56} & -S_{56} & 0 & 0 & 0 & -S_7 & C_7 S_8 \\ 0 & j_{22} & -S_4 S_{56} & -C_{56} & 0 & 0 & -1 & 0 & -C_8 \\ 0 & S_3 S_4 & C_4 & 0 & 1 & 1 & 0 & C_7 & S_7 S_8 \\ j_{41} & j_{42} & j_{43} & -d_6 C_{56} & d_7 + a_5 S_6 & d_7 & 0 & 0 & 0 \\ j_{51} & j_{52} & j_{53} & d_6 S_{56} & a_6 + a_5 C_6 & a_6 & 0 & 0 & 0 \\ j_{61} & C_3 A_2 + S_3 A_1 & S_4 A_1 & A_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (C.2)$$

$$j_{12} = -S_3 C_4 C_{56} - C_3 S_{56}$$

$$j_{22} = S_3 C_4 S_{56} - C_3 C_{56}$$

$$j_{41} = S_2(C_3 C_4 C_{56} - S_3 S_{56}) + C_2 S_4 C_{56}$$

$$j_{51} = -S_2(C_3 C_4 S_{56} + S_3 C_{56}) + C_2 S_4 S_{56}$$

$$j_{42} = S_3 S_4 A_3 + d_6 j_{22}$$

$$j_{52} = -S_3 S_4 A_4 - d_6 j_{12}$$

$$j_{43} = C_4 A_3 - d_6 S_4 S_{56}$$

$$j_{53} = -C_4 A_4 - d_6 S_4 C_{56}$$

$$A_1 = d_4 - a_5 S_5 - a_6 S_{56} + d_7 C_{56}$$

$$A_2 = a_5 C_5 + a_6 C_{56} + d_7 S_{56}$$

$$A_3 = d_4 C_{56} + a_5 S_6 + d_7$$

$$A_4 = d_4 S_{56} - a_5 C_6 - a_6$$

APPENDIX D

The Augmented Jacobian of the CIRSSE manipulator

The augmented Jacobian J_a is defined by:

$$J_a = \begin{bmatrix} J_e \\ dg/dq \end{bmatrix} \quad (D.1)$$

where dg/dq is a $(3 \times n)$ matrix that represents the redundant task space velocities. The kinematic functions of the Puma-Positioner are identity functions. The redundant task space velocities are just the velocities of joints one, two and three and are described by:

$$\dot{q}_1 = \dot{q}_1 \quad (D.2)$$

$$\dot{q}_2 = \dot{q}_2 \quad (D.3)$$

$$\dot{q}_3 = \dot{q}_3 \quad (D.4)$$

The kinematic functions of the Puma-Platform are $d1$, he and $theta$. Their velocities, $\dot{d1}$, \dot{he} and \dot{theta} , are described by:

$$\dot{d1} = \dot{q}_1 \quad (D.5)$$

$$\begin{aligned} x_1 &= a_5^2 + a_6^2 + d_7^2 + 2a_5(d_7 S_6 + a_6 C_6) \\ \dot{he} &= \frac{((a_6^2 + d_7^2 - a_5^2)^2 - x_1^2) \dot{q}_6}{2x_1^{\frac{3}{2}}} \end{aligned} \quad (D.6)$$

$$\dot{theta} = \frac{p2ey * p2ez - p2ez * p2ey}{(p2ey)^2 + (p2ez)^2} \quad (D.7)$$

$$p2ey = -c_1 * s_1 * pex - s_1 * s_2 * pey + c_2 * pez$$

$$p2ez = s_1 * pex - c_1 * pey$$

$$c_1 = \frac{px}{\sqrt{px^2 + py^2}}$$

$$s_1 = \frac{py}{\sqrt{px^2 + py^2}}$$

$$c_2 = \frac{\sqrt{px^2 + py^2}}{\sqrt{px^2 + py^2 + pz^2}}$$

$$s_2 = \frac{pz}{\sqrt{px^2 + py^2 + pz^2}}$$

LITERATURE CITED

- [1] Y. Nakamura and H. Hanafusa, "Optimal redundancy Control of Robotic Manipulators", *Internat. Journal of Robotics Research*, vol 6, no 1, pp 32-42, 1987
- [2] J. Hollerbach and K.C. Suh, "Local vs Global Torque Optimization of Redundant Manipulators", *Proc. Int. Conf. on Robotics and Automation*, Raleigh, N.Carolina, pp 619-624, 1987
- [3] C.A. Klein and C.H. Huang, "Review of Pseudo-inverse control for use with kinematically redundant manipulators", *IEEE Trans. on Sys,Man and Cybernetics*. vol SMC-13-3, pp245-250, 1983
- [4] J.M. Hollerbach and K.C. Suh, "Redundant resolution of manipulators through torque optimization", *IEEE Conf. on Robotics and Automation*, Mar 25-28, St. Louis. MO. pp 1016-1021. 1985
- [5] D.E. Whitney, "Resolved motion rate control of manipulators and human prosthesis", *IEEE Trans on Man,Machine and Systems*, Vol MMS-10-2, pp 57-53, 1989
- [6] H. Zghal, R.V. Dubey and J.A. Euler, "Efficient Gradient Projection Optimization for redundant manipulators", *IEEE Int. Conf. on Robotics and Automation*. pp 1000-1010. 1990
- [7] H. Zghal, R.V. Dubey and J.A. Euler, "Efficient Gradient Projection Optimization for a 7 dof manipulator", *IEEE Int. Conf. on Robotics and Automation*. pp 28-36, April 1988
- [8] Sukhan Lee and Jang M. Lee, "Multiple task point control of a redundant manipulator", *IEEE Int. Conf. on Robotics and Automation*, pp 988-993 , 1990
- [9] Lorenzo Sciavicco and Bruno Siciliano, "Solution Algorithm to the inverse kinematic problem for redundant manipulators", *IEEE Int. Conf. on Robotics and Automation*. pp 403-410, 1988
- [10] Lorenzo Sciavicco and Bruno Siciliano, "On the use of Redundancy in robotic kinematic control", *IEEE Int. Conf. on Robotics and Automation*, pp 1370-1375, 1988
- [11] Homayoun Seraji, "Configuration control of redundant manipulators", *IEEE Int. Conf. on Robotics and Automation*, pp 472-490, Vol 5,4 , Aug 1989

- [12] Charles Wampler II, "The inverse function approach to kinematic control of a redundant manipulator", IEEE Int. Conf. on Robotics and Automation, pp 1364-1369
- [13] J.M. Hollerbach, "Optimum kinematic design for a 7 dof manipulator", Robotics Research, MIT Press, pp215-222, 1985
- [14] T. Shamir, "Remarks on some dynamic problems of controlling redundant manipulators", IEEE Trans. on Automatic Control, Vol 35,3, pp 341-344, 1990
- [15] Nazareth S. Bedrossian, "Classification of Singular Configurations for Redundant Manipulators", IEEE Int. Conf. on Robotics and Automation, pp 818-823, 1990
- [16] J. J. Craig, "Introduction to Robotics, Mechanics and Control ", Addison-Wesley, Reading, Mass., 1986
- [17] Amir Fijany and Antal K. Bejczy, "Efficient Jacobian Inversion for the control of simple robot manipulators", IEEE Proc. on Int. Conf. of Robotics and Automation. April 24-29, pp 999-1007, 1988